Click Here

David J. Barnes & Michael Kölling Objects First with Java A Practical Introduction using BlueJ Seventh Edition, Pearson, 2016 ISBN (US edition): 978-0-138-087173 ISBN (Global Edition): 978-1-924-63599 These are the instructions for the generic BlueJ installer (the installer "for other operating systems"). This installer can be used on most Unix-like operating systems as well as Mac OS X and Windows. However, it is generally preferable to use the "native" installer when available. The generic installer requires that a Java JDK be installed separately. There are two possibilites depending on the version of BlueJ: For BlueJ 5.4.0 and later, OpenJDK 21 and OpenJFX 21 are required, which are only available for 64-bit operating systems. You can download the OpenJDK from Adoptium: choose version "21 - LTS" and then your operating system and architecture. You also need OpenJFX 21 -- download the "SDK" download for your operating system from Gluon. This is a zip which you should unzip locally. The BlueJ installer will ask you for the path of the JDK and unzipped JavaFX. For BlueJ 5.1.0 and later, OpenJDK 17 and OpenJFX 17 are required, which are only available for 64-bit operating systems. You can download the OpenJDK from Adoptium: choose version "17 - LTS" and then your operating system and architecture. You also need OpenJFX 17 – download the "SDK" download for your operating system from Gluon. This is a zip which you should unzip locally. The BlueJ installer will ask you for the path of the JDK and unzipped JavaFX. For BlueJ 5.0.0 and later, OpenJDK 11 and OpenJFX 11 are required, which are only available for 64-bit operating systems. You can download the OpenJDK from the official site, which is a .zip which you must unzip locally. You also need OpenJFX 11 – download the SDK download for your operating system from Gluon. Again, this is a zip which you should unzip locally. The BlueJ installer will ask you for the path of the unzipped directories. For BlueJ 4.0.0 to 4.1.4 a Java 8 JDK is required, and we recommend installing the latest JDK 8 update -- you can download the Oracle JDK from the Oracle website. Make sure to get the JDK (not the JRE) and to download the correct version. Installation Procedure Download the installer, making sure to keep track of where it is in the filesystem. Start your Command Prompt or Terminal application. You will complete the installation by typing some commands into this application. (For Windows users, see these instructions for how to start the Command Prompt). Run the following command: java -jar ... replacing with the correct path to the installer .jar file that you downloaded. If you receive a "command not found" or similar error, you may need to specify the full path to the Java executable. For example: "C:\Program Files\Java\jdk1.8.0_144\bin\java.exe" -jar "C:\Users\SarahJ\Downloads\bluej-411.jar" (this example is for Windows; other operating system paths normally use '/' as a path separator). The installer should start and present a window where you can specify both the path to the JDK, and the path to OpenJFX, (make sure it is correct!) and where you want BlueJ to be installed. It is usually best to install BlueJ somewhere under your home folder. Take note of the path as you will need it to run BlueJ later. Running BlueJ To run BlueJ: On Windows: Start the Command Prompt application. Change into the directory where you installed BlueJ: cd... replacing with the path you chose to install BlueJ to. Start the BlueJ batch file: bluej.bat On other operating systems: Start the Terminal application. Change into the directory where you installed BlueJ: cd ... replacing with the path you chose to install BlueJ to. Start the BlueJ shell script file: ./bluej Support Please see the support links on the main page. Objects First with Java A Practical Introduction using BlueJ Chapter sequence Chapter 1 Objects and Classes Chapter 2 Understanding Class Definitions Chapter 3 Object Interaction Chapter 4 Grouping Objects Chapter 5 Functional Processing of Collections Chapter 6 More-Sophisticated Behavior Chapter 7 Fixed-Size Collections—Arrays Chapter 8 Designing Classes Chapter 9 Well-Behaved Objects Chapter 10 Working in Teams Chapter 11 Recursion Chapter 12 Improving Structure with Inheritance Chapter 13 More About Inheritance Chapter 14 Further Abstraction Techniques Chapter 15 Data-Oriented Programs Chapter 16 A Brief History of Java Chapter 17 Building Graphical User Interfaces Chapter 18 Handling Errors Chapter 19 Designing Applications Chapter 20 A Case Study Back to book description Back to main page Reference/Tutorials | Extensions | FAQ and Support | Publications Please note that some of the written material is out-of-date. We are currently in the process of producing updated reference material. Unit testing Teamwork Raspberry Pi BlueJ's functionality can be enhanced using a variety of available extensions. The FAQ — frequently asked questions. Problems commonly encountered when using BlueJ and how to solve them. Bug Database — known bugs in BlueJ. For problems not found in either of the above: Support Request Form — ask for help if you have problems with BlueJ. A number of journal and conference papers about BlueJ have been published. For details, see the publications page. The following is a list of BlueJ-related publications, which give insights into the design and development of BlueJ and computer science education in general: [PDF] Kouznetsova, S., Using BlueJ and Blackjack to teach object-oriented design concepts in CS1, Journal of Computing Sciences in Colleges, Volume 22 Issue 4, April 2007. This paper describes a course that uses an implementation of the card game 'BlackJack' with BlueJ as an introduction to OO programming (presented at CCSC: Central Plains 2007 conference, April 2007) [PDF] Van Haarlen, K. and Hagan, D., Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool, Information Science + Information Technology Education Joint Conference, Rockhampton, QLD, Australia, June 2004. [PDF] Kölling, M., Quig, B., Patterson, A. and Rosenberg, J., The BlueJ system and its pedagogy, Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, Vol 13, No 4, Dec 2003. This is a general BlueJ overview paper. Something we should have written long ago, but didn't. It explains some of the basic thoughts behind BlueJ, and some recent developments. [PDF] Patterson, A., Kölling, M. and Rosenberg, J., Introducing Unit Testing With BlueJ, Proceedings of the 8th conference on Information Technology in Computer Science Education (iTiCSE 2003), Thessaloniki, 2003. This paper describes the idea behind JUnit integration in BlueJ, its design and the effect for the user. [PDF] Kölling, M. and Rosenberg, J., Guidelines for Teaching Object Orientation with Java, Proceedings of the 6th conference on Information Technology in Computer Science Education (iTiCSE 2001), Canterbury, 2001. A few hints about how to design assignments for introducing Java programming (with our without BlueJ). These are some general thoughts - this is then presented in more concrete terms in The BlueJ system and its pedagogy above. [PDF] Kölling, M., Teaching Object Orientation with the Blue Environment, Journal of Object-Oriented Programming, Vol. 12. No 2, 14-23, 1999. [PDF] Kölling, M., The Design of an Object-Oriented Teaching Environment and Language for Teaching, PhD Thesis, Basser Department of Computer Science, University of Sydney, 1999. [PDF] Kölling, M., The Blue Language, Journal of Object-Oriented Programming, Vol. 12 No. 1, 10-17, 1999. [PDF] Kölling, M., The Problem of Teaching Object-Oriented Programming, Part 2: Environments, Journal of Object-Oriented Programming, Vol. 11 No. 9, 6-12, 1999. [PDF] Kölling, M., The Problem of Teaching Object-Oriented Programming, Part 1: Languages, Journal of Object-Oriented Programming, Vol. 11 No. 8, 8-15, 1999. [PDF] Kölling, M. and Rosenberg, J., Support for Object-Oriented Testing, in Technology of Object-Oriented Languages and Systems (TOOLS) 28, IEEE, Melbourne, 204-215, 1998. [PDF] Rosenberg, J. and Kölling, M., I/O Considered Harmful (At least for the first few weeks), in Proceedings of the Second Australasian Conference on Computer Science Education, ACM, Melbourne, 216-223, July 1997. [PDF] Rosenberg, J. and Kölling, M., Testing Object-Oriented Programs: Making it Simple, in Proceedings of 28th SIGCSE Technical Symposium on Computer Science Education, ACM, San Jose, Calif., 77-81, February 1997. [PDF] Kölling, M. and Rosenberg, J., An Object-Oriented Program Development Environment for the First Programming Course, in Proceedings of 27th SIGCSE Technical Symposium on Computer Science Education, ACM, Philadelphia, Pennsylvania, 83-87, March 1996. [PDF] Kölling, M. and Rosenberg, J., Blue - A Language for Teaching Object-Oriented Programming, in Proceedings of 27th SIGCSE Technical Symposium on Computer Science Education, ACM, Philadelphia, Pennsylvania, 190-194, March 1996. [PDF] Kölling, M., B. Koch and Rosenberg, J., Requirements for a First Year Object-Oriented Teaching Language, in ACM SIGCSE Bulletin, ACM, Nashville, 173-177, March 1995. BlueJ now supports Git repositories as part of its teamwork features. This tutorial briefly explains how to use the new Git support in BlueJ. Contents Introduction to version control and Git It is important to learn some basic Git terminology: Working copy: The place where your BlueJ's shared project is located. Whenever you make any changes to your files in your shared BlueJ project, you are making those changes in the working copy. Local repository: A local Git repository is stored in the same directory as the BlueJ project itself. It is a subdirectory called '.git' that contains all the files needed in order to perform the version control operations (keep track of files changes, deletions, etc.). You should not touch this directory. Remote repository: A remote Git repository is used to keep track of your shared BlueJ project, but resides in another computer. It is usually used to share code with other developers: you can push your changes to the remote repository, and you can pull changes from it. Creating a Git repository There are several web-based Git repository hosting service. They can be a good option if you want to very quickly and easily create repositories to host your BlueJ projects. We talk about some of the most popular web-based Git repository hosting services below. GitHub GitHub is one of the most popular a web-based Git repository hosting service. It provides a web-based graphical interface where you can see your repository, its files, commits, etc. It also provides forums where you can discuss features in your projects, and a wiki engine where you and your collaborators can create documentation for your project. In order to use GitHub, you should create a completely empty repository. You can find a tutorial on how to create your GitHub repository here. Important: in step 5, UNTICK "Select Initialize this repository with a README", so that you get an empty repository. After your create a new repository, GitHub shows a screen with the typical usage of the repository using the Git command line commands (Figure 1), but this should be ignored. BlueJ will perform them automatically when checking out a project. Make note of the repository address (below the phrase: "Quick setup - if you've done this kind of thing before"): you will need to use it together with your GitHub username and password to share a project into or checkout a project from this repository (see the instructions in the following sections). Figure 1: After creating a new repository in GitHub: quick setup screen To checkout the project, go to the menus Tools -> Team -> Share this project. Note: since BlueJ uses username and password authentication. Bitbucket Bitbucket is a web-based Git repository hosting service, similar to GitHub. One of it's differences from GitHub is that you can create free private repositories limited to 5 users. You can find a tutorial on how to create a Git repository in Bitbucket here. Given you have GitLab installed in your local network. You can find a tutorial on how to create a Git repository here. Sharing a BlueJ project using Git Now you have your empty Git repository set, you may want to share your BlueJ project with other users. In order to do so, open the existing project you want to share. Select: Tools -> Team -> Share this project menu (Figure 2). You should see a dialog similar to the one in Figure 3. Figure 2: Sharing a project. Select "Git" as your server type. You should now enter: Your name: The name associated to the changes you make to the repository. Your e-mail: The e-mail associated to the changes you make to the repository. User: The username used to access the remote repository. This is usually your login in the remote repository, and it is usually case sensitive. Password: The password associated to the username. Repository URI: A URI containing the address of your Git repository containing the BlueJ project. Figure 3: Settings for sharing/checking out a project with Git. Tip: It is good practice to click on the "Check connection" button: this button tests if BlueJ can connect to the remote repository and if the URI does contain a valid Git repository. Opening a project from a Git repository in BlueJ In order to open a project previously shared on a remote Git repository, you need to create a local clone of the remote Git repository (this operation should be done only once). This local clone of the project can later be open and used just like a normal BlueJ project. To create the local clone of the project in BlueJ you need to execute the command "Check out project". It can be accessed by going to the menus: Tools->Team->Checkout project... Then a dialog similar to the one in Figure 3 will appear. Note: The username in BlueJ is actually performing a 'clone' operation to create a local copy of the repository, with the main branch checked out into the working copy. You should select "Git" as your server type. This will allow you to enter: Your name: The name associated to the changes you make to the repository. Your e-mail: The e-mail associated to the changes you make to the repository. User: The username used to access the remote repository. This is usually your login in the remote machine, and it is usually case sensitive. Password: The password associated to the username. Repository URI: A URI containing the address of your Git repository containing the BlueJ project (ex: . Tip: Click on the "Check connection" button: this button tests if BlueJ can connect to the remote repository and if the URI does contain a valid Git repository. After clicking on "OK", you should now select where you want BlueJ to checkout the project in your machine. You should choose a name for your project. The project name will be created with the name of the following Remote status: Blank: The file is up to date in the remote repository and no changes should take place. Checking the project status Once you have a shared project, it is likely that either you or other users made changes to the project, therefore, it is important to know if you have changes that are ready to commit to your local repository, and if there are changes already made in the remote repository (push), or if you need to update your local project because someone made a change in the remote repository. This information is presented on the status dialog, which can be displayed by pressing the "Status" button on BlueJ main screen (Figure 4). Figure 4: BlueJ main screen status button Figure 5: Status dialog with local changes. The Figures 5 and 6 have three columns: File name: The files that are being tracked by the version control. These are the project .java files, the project's README.TXT file, any subfolders and the Diagram layouts file BlueJ special file that helps drawing the class diagrams). Local status: The status of the files in the working copy relative to the local repository. Remote status: The status of the listed files relative to remote repository: in Figure 9, we can see that a push action would create "NewClass.java", modify "README.TXT" and remove "Class2.java". A file can have one of the following Local status: Blank: The file is up to date in the local repository and no changes should take place. New: The file is new in the working copy. A commit would make Git track the file and add it to the local repository. Modified: The file exists in the local repository, and it is always in conjuction to one of the previous status, indicating which effect a pull would make in the remote repository. Needs push: This status can only appear in the remote repository, and it is always in conjuction to one of the previous status, indicating which effect a pull would make in the remote repository. Needs update: This status can only appear in the remote repository, and indicates that your local file is older in comparison to the one in the remote repository: you need to perform an update. Figure 6: BlueJ Status dialog with changes to be pushed. Committing and pushing changes Commit As you make changes in your project, you may want to put those changes under version control. This way you may share those changes with other users, by pushing changes into the remote repository. In Figure 7, we show the commit/push window. In the window we can see a list of files ready to commit: These files have modified in some way since the last time an update was made: you may have delete some of them (Class2.java); or created new ones (NewClass.java). You will need to write a comment in your commit: what are the new changes that you made in this commit? this is important as a way of describing what these changes do. This comment will be written in the repository log and are a useful way of quickly and clearly communicating your changes to other users. The commit is done by pressing the Commit button. You don't need to be connected in order to make commits: Git is a distributed version control system, which means that you have a fully functional local repository inside your BlueJ project, and when you commit you are actually putting your changes to this local repository. Push After commiting you changes to your local repository, they are ready to be pushed to the remote repository, where they can be shared by the other users. You can see the list of files ready to be pushed in Figure 5. Figure 8: Pushing changes into the remote repository. You need to be able to connect to the remote repository when you push. BlueJ may ask for the password to connect to your remote repository. Checking project history BlueJ offers a way of looking into the repository log where you can see the commit messages, files filters at the end of the window: "Show file" where we display only the commits that contains the selected file, and "Show user", where you can see the only the commits of the selected user. Updating your project Pressing the "Update" button on the main screen will open the update window (Figure 10), there you will see the list of files affected by this update. To update, you should press the "update" button. Figure 10: The Update project window. Support Please see support page. Objects First with Java A Practical Introduction using BlueJ This Document describes the goals, contents and main pedagogical concepts of this book. It is largely identical with the preface in the book. Contents of this description: Java BlueJ Real objects first An iterative approach Project-driven approach Concept sequence rather than language constructs Chapter sequence This book is an introduction to object-oriented programming for beginners. The main focus of the book is general object-oriented and programming concepts from a software engineering perspective. While the first chapters are written for students with no programming experience, later chapters are suitable for more advanced or professional programmers as well. In particular, programmers with experience in a non-object-oriented language who want to migrate their skills into object orienta-tion should also be able to benefit from the book. We use two tools throughout the book to enable the concepts introduced by the text in our practice: the Java programming language and the Java development environment BlueJ. Java was chosen because of both its language design and its popularity. The Java programming language itself provides a clean implementation of most of the important object-oriented concepts, and serves well as an introductory teaching language. Its popularity ensures an immense pool of support resources. In any subject area, having a variety of sources of information available is very helpful, for teach-ers and students alike. For Java in particular, countless books, tutorials, exercises, compilers, en-vironments, and quizzes already exist, in many different kinds and styles. Many of them are online and many are available free of charge. The huge amount of high quality support material makes Java an excellent choice as an introduction to object-oriented programming. With so much Java material already available, is there still room for more to be said about it? We think there is, and the second tool we use is one of the reasons . . . BlueJ BlueJ deserves some comment. This book is unique in its completely integrated use of the BlueJ environment. BlueJ is a Java development environment that is being developed and maintained by the Programming Education Tools Group at King's College London, UK, explicitly as an environment for teaching introductory object-oriented programming. It is better suited to introductory teaching than other environments for a variety of reasons: The user interface is much simpler. Beginning students can typically use the BlueJ environ-ment in a competent manner after 20 minutes of introduction. From then on, instruction can concentrate on the important concepts at hand—object orientation and Java—and no time needs to be wasted talking about environments, file systems, class paths, or DLL conflicts. The environment supports important teaching tools not available in other environments. One of them is visualization of class structure. BlueJ automatically displays a UML-like diagram rep-resenting the classes and relationships in a project. Visualizing these important concepts is a great help to both teachers and students. It is hard to grasp the concept of an object when all you ever see on the screen is lines of code! The diagram notation is a simple subset of UML, tailored for beginning students. This makes it easy to understand, but also allows migration to full UML in later courses. One of the most important strengths of the BlueJ environment is the user's ability to directly create objects of any class, and then to interact with their methods. This creates the opportunity for direct experimentation with objects, with little overhead in the environment. Students can almost "feel" what it means to create an object, call a method, pass a parameter, or receive a re-turn value. They can try out a method immediately after it has been written, without the need to write test drivers. This facility is an invaluable aid in understanding the underlying concepts and language details. BlueJ includes numerous other tools and characteristics that are specifically designed for stu-dents of software development. Some are aimed at helping with understanding fundamental concepts (such as the scope highlighting in the editor), some are designed to introduce addi-tional tools and techniques, such as integrated testing using JUnit, or teamwork using a ver-sion control system, such as GIT, once the students are ready. Several of these features are unique to BlueJ. BlueJ is a full Java environment. It is not a cut-down, simplified version of Java for teaching. It runs on top of the OpenJDK Java Development Kit, and makes use of the standard compiler and virtual machine. This ensures that it always conforms to the official and most up-to-date Java specification. The authors of this book have many years of teaching experience with the BlueJ environment (and many more years without it before that). We both have experienced how the use of BlueJ has in-creased the involvement, understanding, and activity of students in our courses. One of the au-thors is also the development head of the BlueJ system. Real objects first Object orientation has become the most important implementation technique of today's software developers. Unfortunately, the Java language does not make this noble goal very easy. Numerous hurdles of syntax and detail have to be overcome before the first experience with a living object arises. The minimal Java program to create and call an object typically includes writing a class; writing a main method, including concepts such as static methods, parameters, and arrays in the signature; a statement to create the object ("new"); an assignment to a variable; the variable declaration, including variable type; a method call, using dot notation; possibly a parameter list. As a result, most textbooks typically either have to work their way through this forbidding list, and only reach objects on page 80 or chapter 5; or use a "Hello, world!"-style program with a single static main method as the first example, thus not creating any objects at all. With BlueJ, this is not a problem. A student can create an object and call its methods as the very first activity! Because users can create and interact with objects directly, concepts such as classes, objects, methods, and parameters can be discussed in a concrete manner before looking at the first line of Java syntax. Instead of explaining more about this here, we suggest that the cur-cus-reader dip into Chapter 1—things will quickly become clear then. An iterative approach Another approach discussed aspect of this book is that it follows an iterative style. In the computing educa-tion community, a well-known educational design pattern exists that states that important con-cepts should be taught early and often. It is very tempting for textbook authors to try and say everything about a topic at the point where it is introduced. For example, it is common, when in-troducing types, to give a full list of built-in data types, or to discuss all available kinds of loop when introducing the concept of a loop. These two approaches conflict: we cannot concentrate on discussing important concepts first, and at the same time provide complete coverage of all topics encountered. Our experience with text-books is that much of the detail is initially distracting, and has the effect of drowning the the important points, thus making them harder to grasp. In this book we try to cover all the important topics several times, both within the same chapter and across different chapters. Concepts are usually introduced at a level of detail necessary for un-derstanding and applying to the task at hand. They are revisited later in a different context, and understanding deepens as the reader continues through the chapters. This approach also helps to deal with the frequent occurrence of mutual dependencies between concepts. Some teachers may not be familiar with the iterative approach. Looking at the first few chapters, teachers used to a more sequential introduction will be surprised about the number of concepts touched on this early. It may seem like a steep learning curve. It is important to understand that this is not the end of the story. Students are not expected to un-derstand everything about these concepts immediately. Instead, these fundamental concepts will be revisited again and again throughout the book, allowing students to get a deeper understanding over time. Since their knowledge level changes as they work their way forward, revisiting on patient later allows them to gain a deeper understanding overall. We have tried this approach with students many times. Sometimes students have fewer problems dealing with it than some long-time teachers. And remember: a steep learning curve is not a prob-lem as long as you ensure that your students can climb it! Project-driven approach The introduction of material in the book is project driven. The book discusses numerous pro-gramming projects and provides many exercises. Instead of introducing a new construct and then providing an exercise to put it into practice, we first provide a problem, and then discuss what con-structs are needed to solve it. We state this as a general rule in the chapter introduction. Next: A short description of each chapter See the License for the terms under which the source is distributed. 5.5.0 3 June 2025 Changes in this version: Fixed slowdown bug with Java classes over 5000 lines. Fixed bug with terminal sometimes not showing exceptions if the stderr pane was not previously visible. Fixed bug with rectangle characters showing up in the documentation for classes. Fixed bug with assert panel not changing the OK button to enabled (and the field to disabled) when selecting null/not-null as the assertion while recording a test Show quick fixes even if the locale is one that supports localised Java compiler error messages (e.g. German) The Edit->New CSS File has become Edit->New CSS/Text File and will let you create text files Select several lines in the Java editor, right-click and select "Screenshot line(s)". A screenshot of those lines will be put on the clipboard. The lines will be put on the window.in screen. In the Java editor, middle-click the name of any other class to go to its definition/documentation. (Also available on right-click menu.) When you inspect a String object, view the pane for string content and new "Copy to clipboard" button. New "for-each" button on left side of main window, re-runs last open projects. Slightly increased the space for line numbers to help with four-digit line numbers. Terminal window has new "Always on top" menu option. Made Cmd-comma open preferences on macOS. Ctrl-+ should increase font size on non-QWERTY keyboards. Increased default font size to 12 in new installations. Editor tabs can be dragged to reorder them in the editor window. Ctrl-scrollwheel on Windows and Linux, and two finger pinch-zoom gesture on all OSes changes the editor font size. Changed the default Java editor and terminal font to "Source Code Pro", which supports box-drawing characters. 5.4.2 12 March 2025 Changes in this version: Upgraded jGit library andnchield jGit-SSH library. Fixed various Git issues with SSH. Fixed Git "Check connection" to stop saying connection was fine in some cases when it was not. Changed Git settings dialog to ask for access token instead of password when connecting to GitHub/GitLab. Removed macOS messages from Terminal window. Improved message when you attempt to run the Aarch Mac version on an Intel CPU. 5.4.1 19 September 2024 Changes in this version: Fixed a bug where errors would sometimes not show as red underlines in the editor. The side bar would indicate an error on the line but it was impossible to see the error message. Fixed a bug where the Windows MSI installer would allow multiple per-machine installations of different versions (in the same directory). BlueJ 5.4.1 will correctly remove all prior per-machine installations of earlier BlueJ versions. Fixed a missing message in the Git settings dialog. Fixed a bug where the right-click menu for the Java editor side bar (which let you turn on line numbers) would not appear on Windows. 5.4.0 9 August 2024 Changes in this version: Moved to Java 21, including support for the new language feature switch patterns (including records). Changed Mac build to use the JDK for Apple silicon (M1, M2, etc); Intel build is still available. Added ARM64 Linux build to support Raspberry Pi. Fixed bug with scope highlighting sometimes missing on some lines. Fixed some issues with entering non-English characters (e.g. Chinese, Korean) in Java and Stride. Changed the terminal scope highlighting to be turned off by default. Allowed the terminal tooltip on hover in the terminal. Allowed the terminal scope highlighting to be turned off by default. Added the ability to toggle whether to be printed oddly in the terminal. Fixed a bug which could cause slowdown while a Stride editor was open. 5.3.0 20 March 2024 Changes in this version: Mac Fixed splash screen not showing up until quite late, sped up startup process, fixed application not getting focus after load Stopped Terminal showing blank stderr pane when you run a JavaFX application Added JavaFX doc link to help menu Fixed issue with same-day commits sometimes being sorted wrong in Git history due to sorting by 12-hour clock time Fixed Git history to only show files (instead of other parent (rather than all files present in the repository then) Some slight speed-ups to Java editor performance Added code completion for local variable names (incl parameters, for-each loops and instanceof vars) Added section boundaries in the terminals, and the button which lets you remove all prior terminal content, to group together output from each method calm and added tooltip showing the method call that produced each section (even if you aren't logging method calls) Clear the stderr pane in Terminal on each method call Updated the Git library we use Changed binary files Changed debugger split pane to be 3-way Stopped object highlight enlarging as you step through code Added user-contributed Chinese translations 5.2.1 10 October 2023 A full list of bug fixes can be found on Github. Some highlights: Fixed the Linux installer not working on older Ubuntu and Debian. Fixed the generic JAR installer not working outside Linux. Fixed the JavaFX warning not being suppressed in the terminal in non-English locales. Fixed issue with permits clauses not being parsed correctly when after an extends or implements clause, and messing up dependency calculation. 5.2.0 20 June 2023 A full list of bug fixes can be found on Github. Some highlights: Fixed an issue on Mac where double-clicking a Finder to open a project would load/focus BlueJ but not open the project. Fixed a launch crash on Ubuntu. Fixed a bug where errors on the last line of a Java file (often, reached end of file while parsing after a missing close curly bracket) would not show at all. Removed annoying warning on stdout in the Terminal when running JavaFX programs. You can now focus classes in the class diagram by typing their name (e.g. typing "Tic" will focus "TicketMachine"). Mac is now packaged differently as a DMG. 5.1.0a 27 October 2022 This is a Mac-only release: BlueJ now runs on (and requires) Java/JavaFX 17. Support for all the Java 17 language features has been added to the Java editor. 5.1.0 28 March 2022 In this release: Many different accessibility improvements, especially to the terminal, codepad, but also the general interface. Compiler errors can now be seen in a list pane when accessibility mode is turned on. Fixed a bug in saving the password for Git in the teamwork. Various small editor bug fixes, including one where errors would sometimes not show in the editor. Added System.in/out/err to the code completion dialog as a special case. Stopped passing -source to the compiler by default (this now allows users to specify --release). Cut/copy now do nothing if the selection is empty (previously it would blank the clipboard). Added new method to the plugin API. 5.0.2a 2 December 2021 This is a Mac-only release to address a crash that was happened on MacOS 12 (Monterey). All other operating systems should continue to use 5.0.2. 5.0.2 6 August 2021 In this release: Fixed a problem where BlueJ would crash when launching on some Windows machines. Fixed a bug where using would not work correctly in some programs. Some minor bug fixes to the Java and Stride editors. Fixed a bug where sometimes red error underlines would not show up in the Java editor. Added new methods to the plugin API. A full list of these changes and improvements is available here. 5.0.1 30 April 2021 In this release: Fixed a problem where BlueJ for JUnit 5 and switched to JUnit 5 by default (JUnit 4 is still supported as before). Rewrote the extensions API to work with the new editor. Fixed not being able to alter the password saved for Git in the teamwork. Various small editor bug fixes, including one where errors would sometimes not show in the editor. Added System.in/out/err to the code completion dialog as a special case. Stopped passing -source to the compiler by default (this now allows users to specify --release). Cut/copy now do nothing if the selection is empty. Added new method to the plugin API. 5.0.0 28 January 2021 In this release: Rewrote the Java editor display to be faster and more robust. Added some quick-fixes to the Java editor, suggesting things like adding missing imports. Added support for JUnit 5 and switched to JUnit 5 by default (JUnit 4 is still supported as before). Rewrote the extensions API to work with the new editor. Fixed not being able to alter the password saved for Git in the teamwork. Various small editor bug fixes, including one where errors would sometimes not show in the editor. Added System.in/out/err to the code completion dialog as a special case. The extensions API is planned to change soon, so extension writers will need to make changes. Further details will be available in the coming months. 4.2.1 30 April 2019 In this release: Fixed a startup freeze that could cause BlueJ to be forever stuck on the splash screen in Windows. Fixed a window positioning issue that could cause BlueJ windows to appear off-screen (and thus seem to never appear), especially on Windows. Stopped the "File dependon disk" dialog from popping up again when dismissed, and made it less likely to appear in cases which might be caused by slow network storage. Fixed an issue with not recognising JavaFX installed via Ubuntu package. A full list of fixes and improvements is available here. 4.2.0 7 Feburary 2019 In this release: Moved to Java 11. This means that BlueJ now requires a 64-bit operating system. Added support for the new keyword in Java 10. Fix JUnit Before/Class/AfterClass functionality. Fix various small editor bugs. A full list of fixes and improvements is available here. Note: This is likely to be the last release with support for Subversion. Users are encouraged to migrate to using Git instead. 4.1.4 29 October 2018 In this release: Fixed memory leak in Java editor. A full list of fixes and improvements is available here. Note: This is likely to be the last release with support for Subversion. Users are encouraged to migrate to using Git instead. Note: See Version 4.0.0 notes for supported devices, minimum requirements and major changes since Version 3.1.7 Note: This is the last version of BlueJ supporting Java 8. 4.1.3 7 September 2018 In this release: Fixed various issues with Git support. Many other minor fixes. A full list of fixes and improvements is available here. Note: See Version 4.0.0 notes for supported devices, minimum requirements and major changes since Version 3.1.7 4.1.2 9 November 2017 Major points: Added: Interactive tutorial (under Help -> Interactive Tutorial). Fixed: Many printing problems, especially for the Java editor. Fixed: Stale bugs displayed in editor / compile command ignored. Fixed: Git support working with sub-packages on Windows. A full list of fixes and improvements is available here. Note: See Version 4.0.0 notes for supported devices, minimum requirements and major changes since Version 3.1.7 4.1.1 18 September 2017 Major fixes: Fixed: Occasional exceptions/hangs during Java file editing. Fixed: Many printing problems. Fixed: Class documentation view shows blank on first view. A full list of fixes and improvements is available here. Note: See Version 4.0.0 notes for supported devices, minimum requirements and major changes since Version 3.1.7 4.1.0 23 June 2017 Major fixes: Fixed: Graphical display hang could cause the Java editor and other windows (e.g. Terminal) to turn white and not redraw correctly. Fixed: Several teamwork issues and bugs (especially in Git). Fixed: Several controls (such as buttons in editor) could appear tiny on high-DPI screens on Windows. A full list of fixes and improvements is available here. Note: See Version 4.0.0 notes for supported devices, minimum requirements and major changes since Version 3.1.7 4.0.1 28 March 2017 Major fixes: Fixed: Closing code editor kills created objects and clears the object bench Fixed: After switching to another language, Preferences disappear from the Mac app menu and some keyboard shortcuts will not work Fixed: Debugger is not stopping on some Breakpoints Fixed: Backspace doesn't work in Terminal on Mac Fixed: Add launch crash on Ubuntu Fixed: Various keyboard shortcuts using <extra> markers are broken Fixed: Lack of scrolling with lots of methods either in unit test, Class popup or Object popup menus Fixed [MacOS X]: Projects can't be opened with double click on "package.bluej" when BlueJ is open A full list of fixes and improvements is available here. Note: See Version 4.0.0 notes for supported devices, minimum requirements and major changes since Version 3.1.7 4.0.0 8 March 2017 First feature additions for version 3.0.0 8 October 2009 Summary of most important changes: BlueJ now supports running applets as applications: ...

never terminates if another thread tries to j... The auto-indent should work in both directions There should be an opposite to TAB key indention 1.3.5 11 December 2003 Summary of most important changes: execution speed improvements (using VM optimisation by default) JUnit testing: auto-generated code improved auto-open of last used projects on startup System.exit now restarts VM improvements to extension API standard preference handling on MacOS number of 'recent projects' user definable startup without additional DOS window on MS Windows class diagram supports multiple selection changed inspector look better error messages easier addition of user libraries through use of 'userlib' directory colours for different class types user definable added missing label in some foreign languages 'VM optimize' now available in preferences new menu shortcuts for 'page setup', 'generate documentation' text input wirh multiple projects now works (one terminal per project) editor maintains last display state (interface/implementation) between sessions each package now has separate README note JApplet class now used by default when creating applets instead of Applet 1.3.0 7 August 2003 Summary of most important changes: more robust debugger/execution archtitecture extension API added submission extension added unit test support added improved diagram printing user definable scaling of diagram printout now using jdk 1.4.1 on MacOS if installed Aqua interface improvements on MacOS web browser start problem on Win98/ME resloved EXIT_ON_CLOSE option bug fixed some documentation generation bugs fixed new interface languages available: Portuguese System.err output now print font size user definable printing form feed character (unicode 0x000C) now clears terminal terminal gets automatically cleared on recompile inspection of static fields now in class menu compiler warning display improved 1.2.2 16 October 2002 Summary of most important changes: freeze bug with jdk1.4.1 and debugger fixed 'open recent' option added to Project menu support for 'stand-out' comments System.err output in terminal improved project documentation with spaces in path works now evaluation of expressions added user interface in italian added 1.2.1 1 August 2002 Summary of most important changes: bug with frequent access of floppy drive fixed bracket matching in editor implemented bluej.exe accepts parameters System.err output now directed to BlueJ terminal 'auto-clear' option in bluej terminal new interface languages: Chinese, Czech, Afrikaans 1.2.0 4 April 2002 Summary of most important changes: 326, 335 - support of jdk 1.4 added 285 - return key now activated OK button in parameter dialogs 330 - project documentation fixed with jdk 1.4 343 - improved auto-indentation in editor 347 - last used project directory saved across sessions 000 - added windows installer 1.1.6 16 January 2002 Summary of most important changes: 166 - debugger now displays static variables in a separate list 166 - debugger displays variables when stopped in static method 225 - projects can include project-specific libraries 261 - Frames/JFrames automatically removed when objects are removed 235 - help menu items can be added by users 267 - terminal font size user definable 276 - javadoc can be configured to include private methods 295 - automatic creation of crash recovery files; optional creation of backup files (thanks to David R. Musicant (dmusican@mathcs.carleton.edu) for implementing these features) 241 - proper find/replace implemented in editor 250 - optional line number display in editor 300 - added block indent function to editor 303 - method template for "insert-method" template user definable 310 - japanese localisation added (interface , etc. in japanese); thanks to Runrun 275 - use Command key instead of CTRL as menu shortcut on MacOS 277 - bug fix on MacOS: Show Terminal/Debugger could crash system 298 - extended latin characters (umlaut, accented characters, etc) handled correctly by parser 1.1.5 24 September 2001 Summary of most important changes: 240 - documentation needs update (1.3 req) 12 - June 2001 requires JDK 1.3 or newer uses new (JDK1.3) compiler implementation can get arrays and array element to the object bench local variables of static methods now shown (bug fix) new class templates can be added class templates can be located in arbitrary directory "user home" can be specified display of exceptions in library classes improved bug fixes with editor key definitions new editor functions for cutting/copying/moving lines and words optional soft TABs (tabbing with spaces) in the editor various bug fixes 1.1.3 29 March 2001 can now create objects of library types (eg. String) and call static methods breakpoints in main did not work - fixed editor tab size definable in bluej.defs file syntax colours user definable bug with interface views and multiple drives on Win32 fixed can now re-display project documentation without re-generating debugger "terminate" fixed object inspection from debugger fixed export for packages with sub-packages fixed 1.1.2 9 February 2001 added support for installation on MacOS X interface view of classes changed to display javadoc format in BlueJ editor inspection of elements of large arrays fixed System.exit now terminates all threads bugs with parsing explicit ints fixed more compiler error message help texts added "describe" function (CTRL-D) in editor fixed (had bug on Windows) static data now gets reinitialised with each call to "main(String[])" 1.1.1 1 September 2000 applets are now properly recognised even if not directly inheriting from "Applet" editor key bindings are now properly saved between sessions (JDK 1.3 only) "spinning barber pole problem" (infinitelyspinning barber pole) on Linux fixed classes in the project are now recognised when source is not there (only .class file). This may be used to supply use-only classes to students without source bug with incorrect project closing fixed class positioning when adding class from file corrected long error messages in editor now wrap over two lines. Pressing the "?" button shows the complete message (together with an explanation) default argument for "public static void main(String[])" changed from null to { } to ensure compatibility with DOS editors have icons when minimised 1.1 4 August 2000 UML-style notation for the class diagram support for named and nested packages improved import of non-BlueJ packages terminal text can be saved or copied editor key bindings can be changed (jdk 1.3 only) editor font can be changed "export" function to create standard Java packages and executable jar files projects have project comments (README note) javadoc support added for single classes and whole projects cascading menus for large menus the debugger can now be used with swing applications classes can be renamed in the class source 1.0.3 23 May 2000 static methods on abstract can be called now bug with calling methods after using "Get" has been fixed bug with printing TABs has been fixed printing the class diagram has been improved class names can now be changed by changing the name in the source 1.0.2 8 March 2000 printing class sources has been fixed/improved editor views have been improved bug fixed: end of long lines was not displyed properly bug fix: System.exit() did not work correctly new editor functions: "indent", "break-and-indent", "comment", "uncomment" editor key bindings can be changed (but are not yet saved - they are lost on exit) a default source directory can be defined in the "bluej.defs" file. This serves as a starting point for the file chooser install support for jdk 1.3 added 1.0.1 17 January 2000 installation support for Linux jdk 1.2.2 added preferences dialogue editor font size can be changed non-BlueJ packages can now be opened in BlueJ (with automatic dependency analysis). So far, only packages without nested packages are supported "package..." line automatically fixed when importing classes System.exit() now works for graphical applications bug fixed in inspection of large arrays default parameter "null" added for static void main calls support for internationalisation (so far, German is the only language file supplied besides English) 1.0 23 August 1999 dependencies (eg inheritance relationships) are automatically entered into the source code if entered graphically several bugs in the debugger have been fixed 0.9.7 11 August 1999 text terminal reimplemented; fixed many problems terminal comes to front on output inspect for local variables in debugger implemented dependency display improved: changes in code immediately reflected in diagram 0.9.5 7 July 1999 Implementation changed to run on Java 2 platform Debugger implemented. Includes breakpoints, single stepping, variable inspection Execution implementation changed to use JDI - promises significant improvements in stability and reliability Applets supported (development and viewing in applet viewer and browser) Dependency analysis fixed - works correctly now Many editor improvements Help button for compiler error messages and exceptions Array inspection implemented Exception handling (reporting) improved 0.9.2a 12 May 1999 One bug fix: bluej could not be started on Unix systems without a command line argument. 0.9.2 5 May 1999 internal implementation improvements some editor improvements (key shortcut, tabs, ...) editor function "key bindings" implemented - lists all user functions and their bindings many bug fixes 0.9.1 28 April 1999 printing of source files implemented cleaned up object popup menus classes can no longer be moved out of the screen removed bug in removing arrows backup package file now named "bluej.pkh" instead of "bluej.pkg~" to avoid Windows copying bug 0.9.0 15 March 1999 handling of exceptions: exceptions in user programs are caught and displayed properly System.exit() can now be used normally bug fix: empty string as method result now returned properly bug fix: user defined object as method result now shown correctly bug fixes: names of objects on bench: now ensures that names are unique and valid bug fix: layout of object bench after "get" and "remove" fixed bug fix: objects could not be used properly after put on the object bench through the "Get" operation in the inspect window. This works now. 0.8.9 2 March 1999 Removing of classes has been implemented In the installer, a path to the java executable can be specified and will be used In multi-user system, where the BlueJ home directory is not writable to users, the system should now work much better. The Save As function has been implemented. Many other bug fixes... A free Java Development Environment designed for beginners, used by millions worldwide. Find out more... "One of my favourite IDEs out there is BlueJ"— James Gosling, creator of Java. Created by Supported by Download and Install Version 5.5.0, released 3 June 2025 (Many feature improvements, see more) Note: BlueJ requires a 64-bit operating system, which 95+% of users will have.For 32-bit operating systems, download BlueJ 4.1.4 instead. "Objects First with Java: A Practical Introduction Using BlueJ" is a textbook co-written by the developers of BlueJ and has sold hundreds of thousands of copies worldwide. Our free Blueroom website offers teaching resources and the chance to engage with other educators using BlueJ around the world. Tutorials and reference materials for working in BlueJ. We have several extensions available that add to BlueJ's base functionality.

- fecamu
- http://activesites.atiks.org/files/95564d0b-bc19-4792-bc32-06877e184d4c.pdf
- http://dhins.com/testingsites/advantage_aviation/assets/media/file/1f4a2022-18f9-4ab8-bcea-7b5d2551ff8e.pdf
- what is static electricity in simple words
- unit 9 progress check mcq ap chem
- horrible bosses 2 ending explained
- hebuga
- https://krygina40.ru/UserFiles/File/62081109377.pdf