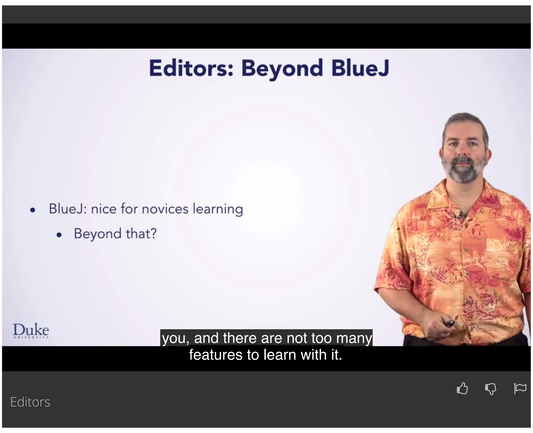
Using Java Beyond BlueJ

- Module Learning Outcomes / Resources 10 min
- Miscellaneous Java 1 min
- Main Method 3 min
- Static 4 min
- Editors 8 min
- Summary 53 sec

Mainstream Java

Review



Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript English

0:03
You've used BlueJ throughout this course because it is a great tool for novices learning to program. It makes things easy for you, and there are not too many features to learn with it. However, as you advance in programming skill, you may want to explore other editors. There are many choices and tradeoffs. Do you prefer simplicity or the power presented by advanced features? As you learn other programming tools, such as revision control systems like Get or debugging tools, how does your tool integrate with them?

0:36
Another important consideration is whether you want to use a graphical user interface, or GUI, which may feel more familiar and comfortable to you, or something where you only use the keyboard. Using only the keys may seem intimidating, but it allows you to work from muscle memory. When you can edit from muscle memory, your editing doesn't interrupt your thoughts about your algorithm design and implementation, which can make it much more efficient.

1:01
As you think about these factors, there is a wide spectrum of editors, ranging from those which are more friendly to novices to those which are more friendly to experts.

1:11
Novice friendly editors, such as BlueJ, are GUI-based and favor simplicity over power.

1:18
As you work your way down the spectrum, you find editors providing more advanced features.

1:24
One popular editor is Eclipse. There are also two very common editors aimed solely at experts, Emacs and Vim. Programmers who use these editors primarily use the keyboard and enjoy the advanced features they provide. However, their power comes with a cost of a learning curve. Emacs is what Drew teaches his students to use in class.

1:46
So, if expert friendly tools, whether editors or other tools, are harder to learn, why would you want to learn them? Learning them is an investment if you plan to use them over the long-term if you look at what you can do with a tool versus the effort you have spent learning it.

2:03
For a novice friendly tool, you can just pick it up and use it, and a do a decent amount with it. As you work with the tool, you can do more, but you quickly plateau in what you can do.

2:13
If you look at an expert friendly tool, you find that, when you start out, it is more difficult to use. You may not be able to do much with it until you invest some effort into learning it. However, as you invest effort into mastering your tool, you surpass what you could do with an novice tool and then benefit from the power of the advanced features. As an analogy, think about recording video. From most people, a novice friendly tool is great. If I'm gonna record a video, I take out my cellphone and press the Record button. It's super easy, and I didn't have to do anything to learn how, but professional videographers use much more sophisticated tools. A high-end professional video camera has a bunch of complicated features, and I wouldn't know how to use one without investing some time in learning it. If I wanted to use any of the advanced features, it would take even more effort. I can stick to my cell phone since I don't need those features, and I'm not a professional in that field. However, for professionals, learning to use the advanced tools is a worthwhile investment. The same principle is true of editors. If you plan to be a casual programmer and just write small programs, a simple editor may be a good choice. However, if you plan to become a serious professional programmer, you will want to have invested the effort in learning an editor whose advanced features will make your job easier in the long run.

3:37
>> Hi, I'm going to walk through using Eclipse, an integrated development environment, or IDE, that's the IDE of choice for many, many Java developers. I've used it to develop most of the material for this course when I'm not using BlueJ, and the UCSD specialization that comes after our specialization also asks learners to use Eclipse. As you know, there's some other IDEs as well, but because I'm most familiar with Eclipse, I'm going to walk through just a few of the features that it has to show you what it's about. I'm using the class Markov runner that we've used before, and I've created Markov2Zero, MarkovOne, and MarkovTwo. We've gone through those in previous lessons. I'm gonna use Eclipse to make MarkovThree, a class that we haven't done yet. We've gone, in previous lessons, from the MarkovTwo to the general Markov model. In Eclipse, I'm coming in just as you do in BlueJ and saying, I need a new Java class. I get a little menu and I'm gonna call this Markov3. The interesting thing about Eclipse is, I can say, here's an interface that I'd like to add, and I'd like to add the IMarkovModel interface. I'm gonna say OK, when I'm finished. Now I have my Markov3 class, and you can see that Eclipse has filled in the stubs for all of the methods I need to implement. They're a part of the IMarkovModel interface. As a reminder, here is the IMarkovModel interface, it has two methods that I must add, SetTraining and getRandomText. We've seen that before, and stub implementation for those were provided in Markov3 by Eclipse. It even has a return value, that's not the right return value, and it does the setTraining method here. The nice part about Eclipse for what we're seeing here is that, for interfaces, Eclipse will fill in the stubs, and then I can copy the code from Markov2 into here and make sure that it runs. I'm not gonna do that now, I just wanted you to see that, with interfaces, that's a nice feature to have. If I started to copy the code in, and I'll just do it a little bit so that you can see what happens. Here's the getRandomText function, I'll just copy the first few lines into my Markov3 class.

6:03
If I'd forgotten for example, in this case Markov2 is gonna be replaced by a 3. We've walked through that in a previous lesson.

6:12
Sb.append, when I type sb. I see a menu pop up of the choices for this method including append. Now BlueJ has the same functionality if you do a Ctrl+Space or right-click+Space. In this case, I'm just going to make sure that I call append properly and put in current, BlueJ also has that functionality. Here, if I say, if I forget the return statement, I get some red Xs. Eclipse is going to complain. In this case, it's told me that it doesn't know where my random is, and it doesn't know what my text are. With these little red Xs, I can do a pop-up and it can say create a local variable or create a field. So I'll create a field myText, and Eclipse came and set this variable myText knowing how it was used here. I can do the same thing here for my random, I will add the field myRandom. It added that up here. It thinks it's an object, it couldn't know that it was a Random. I'm going to change the type to Random, I've got another red X. Import the class Random from java.util. Eclipse knows where this class lives, and so it filled in all this material for me. I'm almost done, but I finally have a red X here because, as it says, if you could look really closely, add a return statement or change the return type to void. I'm missing the return statement, so Eclipse makes a guess and says maybe I should return current. That's not correct, but it's enough to make my program compile, and, as you've discovered, once the program is compiled, I can run it, and that means I can test it and go back and make corrections.

7:56
That takes care of a few of the features that Eclipse allows me to have and make my programming a little more productive by helping me correct my own errors. One more interesting feature that Eclipse can do, I can say I don't like the variable name myText. I'd like to right-click, Refactor, I'd like to Rename this variable. I think myText is not quite right, I'd like to call it myTrainingText.

8:24
When I make that change there, Eclipse has gone through and found all of the occurrences of myText, you can see them highlighted, and they're changed there. Refactoring in Eclipse has lots of many powerful functions that we're not getting into here, but if you continue and do more advanced object-oriented Java programming, you'll see them. Happy programming.

Downloads

- Lecture Video mp4
- Subtitles (English) WebVTT
- Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?