

Applications Programming

Pre-Lab Exercise: Inheritance and NetBeans

Starting from week 8, you will learn to develop GUI applications in Java. This will involve the use of advanced features of Java that go beyond the capabilities of ED and BlueJ.

In this pre-lab exercise, you will learn to use NetBeans, a more powerful IDE (Integrated Development Environment). This is available from the Red Hat applications menu under the Programming sub-menu on all FEIT Linux lab machines. You are also encouraged to install NetBeans on your own PC or laptop. There are two ways to install NetBeans:

1. Download the Java Development Kit version 8 (JDK 8) from <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html> and the latest NetBeans IDE from <https://netbeans.apache.org/download/index.html>.
2. Another option is to download the all-in-one NetBeans + JDK bundle from <https://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-3413139-esa.html>

Make sure to install Java 8 instead of other latest version. This is because the components supporting GUI programming (JavaFX) are removed from the JDK package after Java 11. To support the following GUI programming labs, JavaFX is required to be downloaded and installed separately running on the platform later than Java 8. If you have installed other JDK version other than Java 8, please download and install Java 8 as the primary Java platform used in IDE.

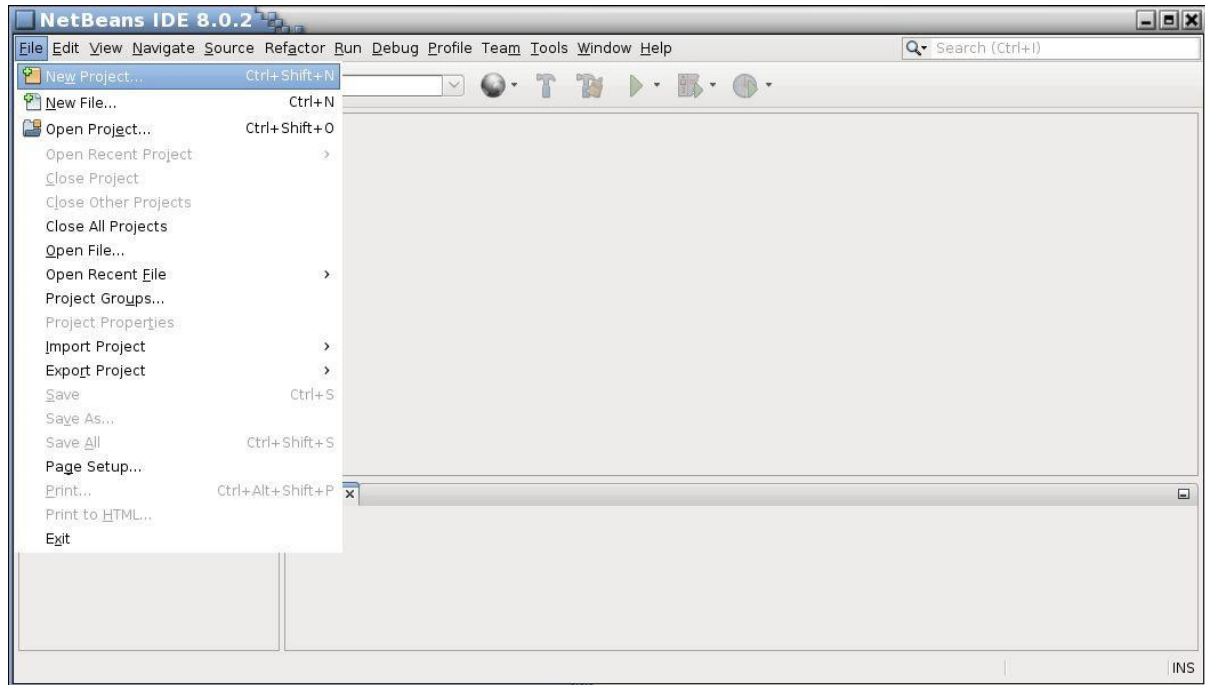
The advantage of downloading the JDK separately is that you will be able to try alternative Java development environments on top of this JDK. It is also important to understand the difference between the JDK (Java Development Kit) and the JRE (Java Runtime Environment). The JDK provides the tools required to develop Java programs such as the compiler. The JRE contains only what is necessary for the end user to run a Java program, so the compiler is not included. As a developer, you need the JDK.

Some students may have already experimented with other non-BlueJ IDEs such as Eclipse, IntelliJ and even programmer's editors such as Vim, Sublime Text and Atom. If you did, that's great! You are free to continue to use your preferred alternative to BlueJ for the rest of semester, provided that it supports Java 8, and that it allows you to create non-Java files within your project, such as XML files, CSS files, JPG files etc.

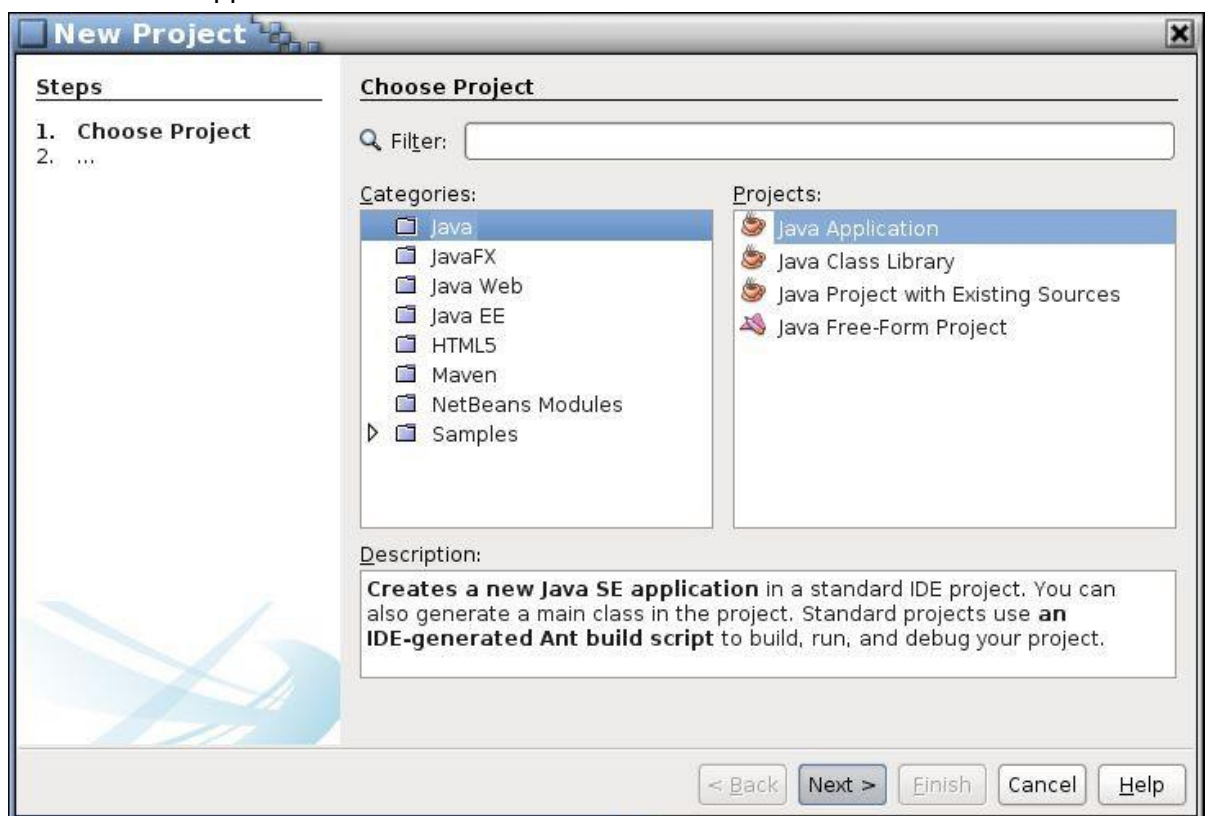
However, for this exercise, the purpose is to become familiar with NetBeans. The following steps show you how to create a project in NetBeans, import files, edit and refactor code, and create a JAR file to submit to Canvas. These instructions assume you are using a Linux workstation in UTS Building 11. Steps may differ slightly if you are using Windows or MacOS.

Importing code into NetBeans

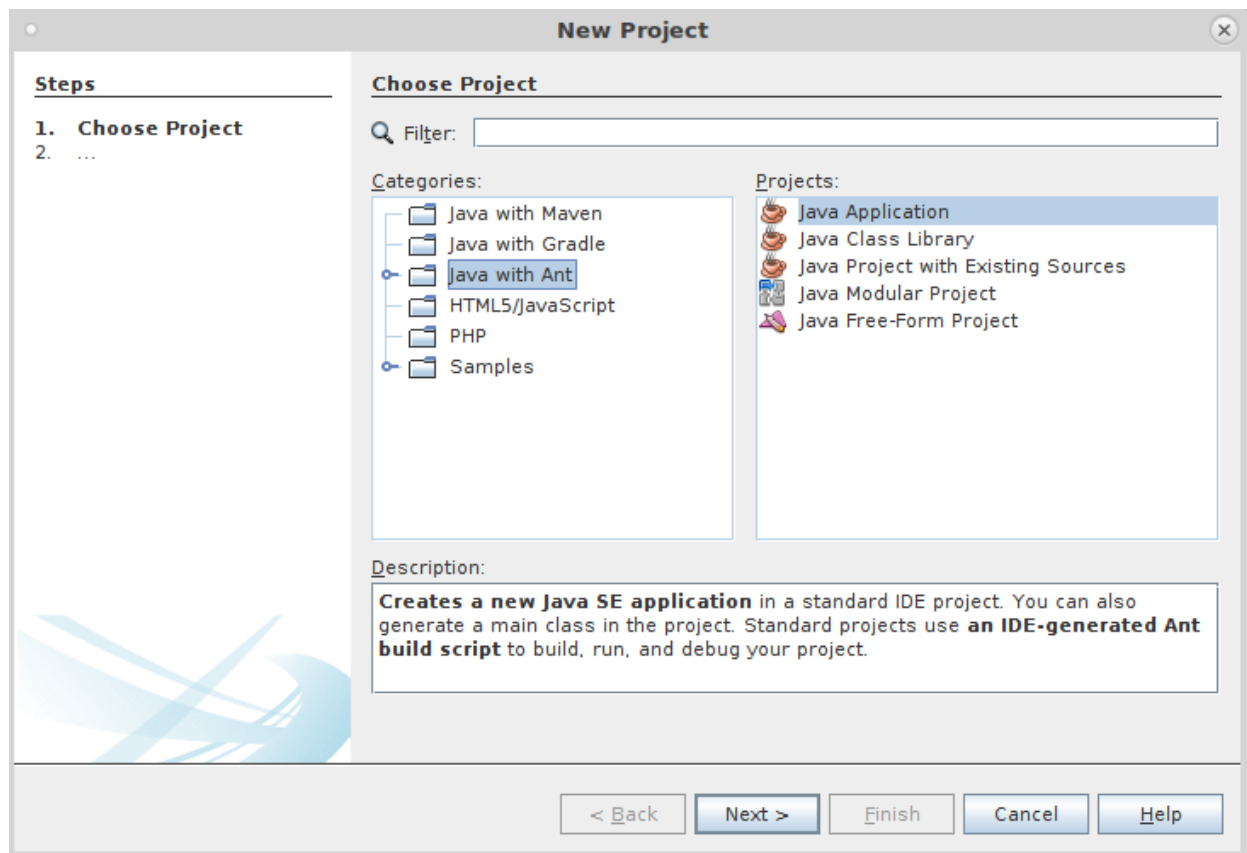
From within NetBeans, create a new project.



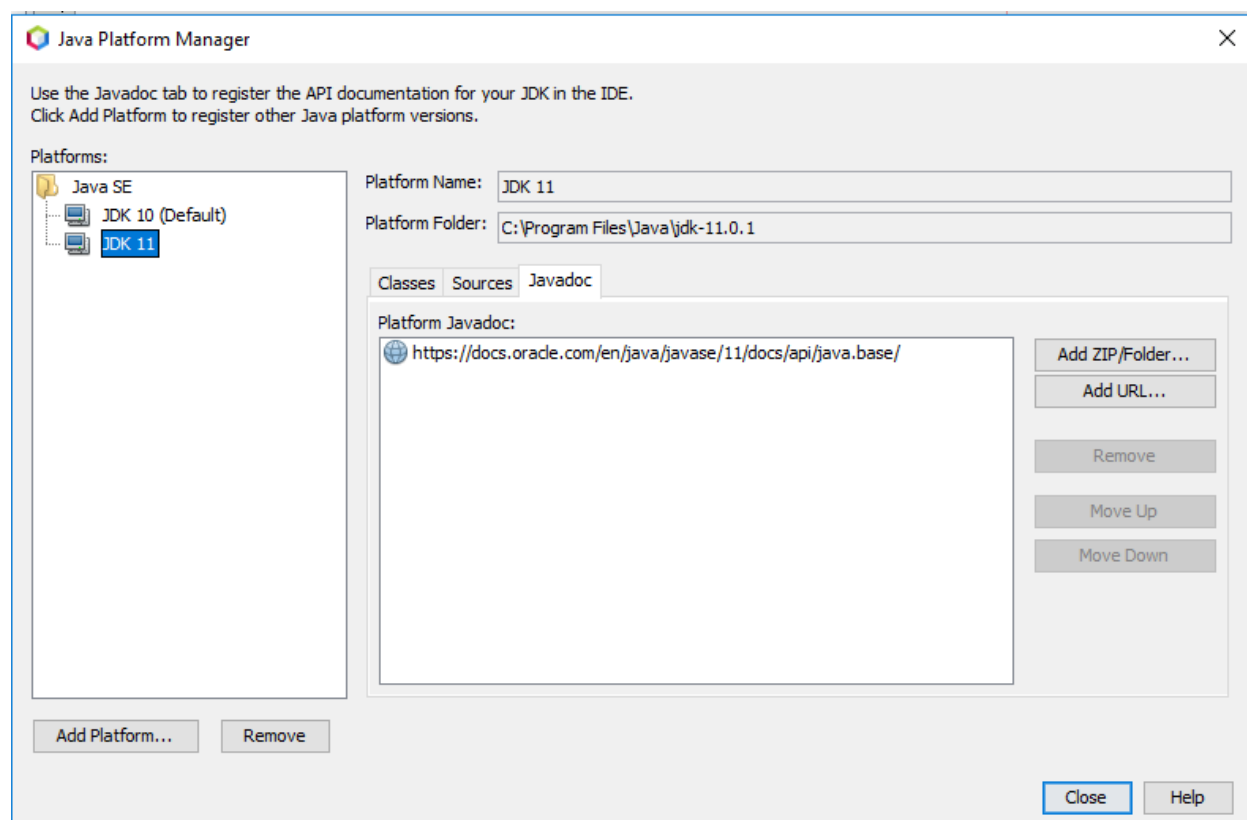
Choose “Java Application”



The latest version of NetBeans IDE will look as follows:
Select Java with Ant and then choose “Java Application”



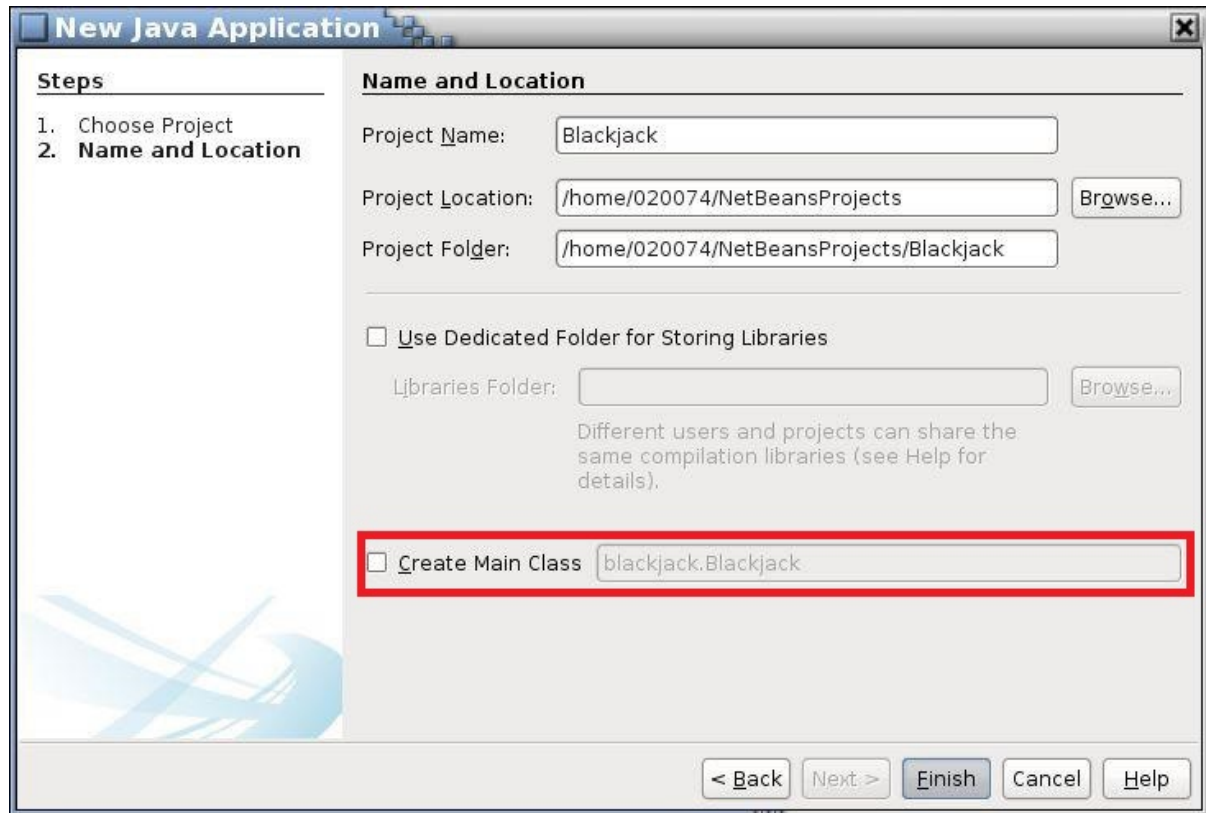
In Java Platform Manager, select “JDK 8”



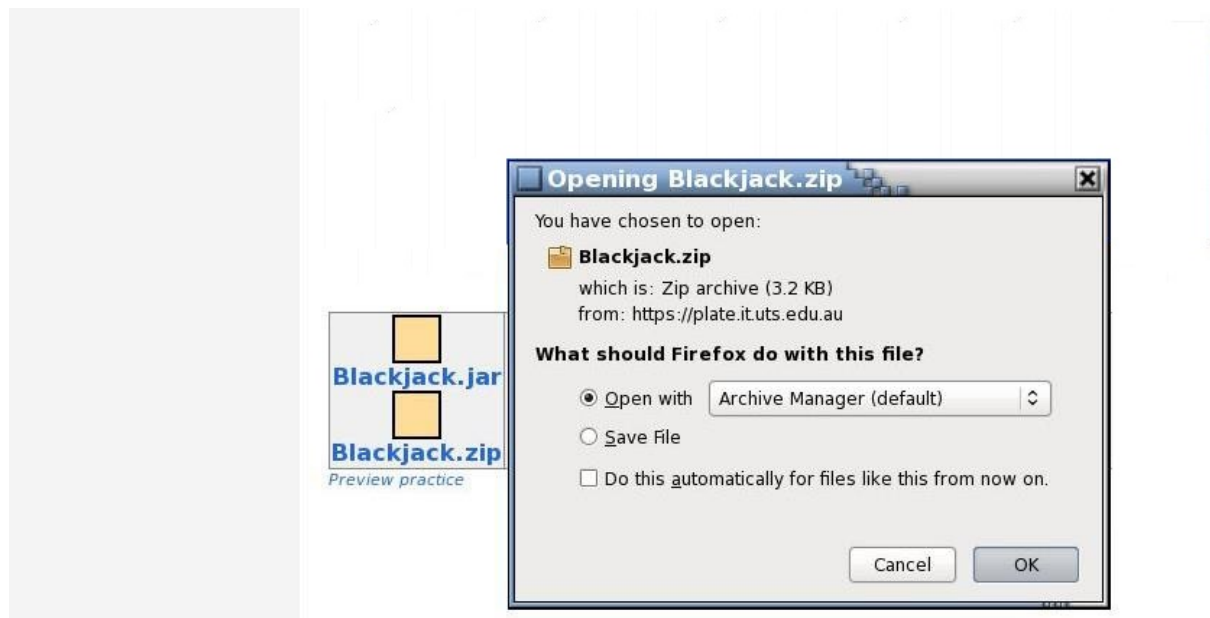
Enter the Project Name: Blackjack

Deselect "Create Main Class". That is, do NOT create a Main Class

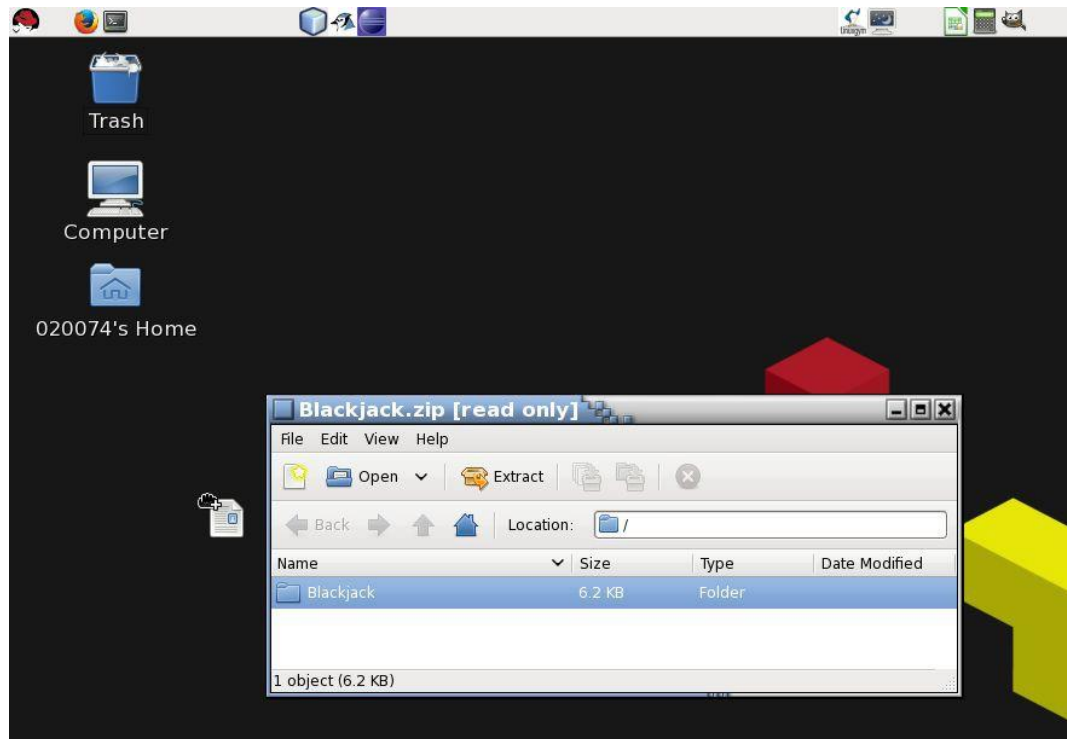
Don't change anything else. Your project location/folder will be automatically chosen



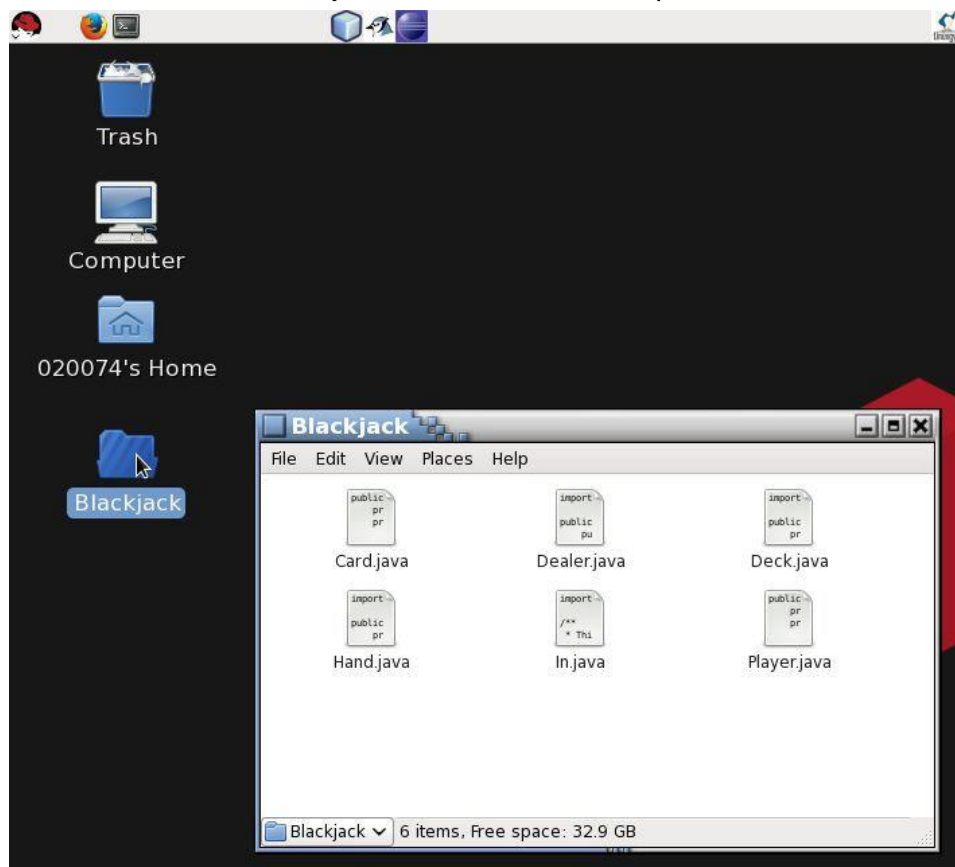
On desktop, click on Blackjack.zip and open it with the Archive Manager (or another preferred application for opening a ZIP file)



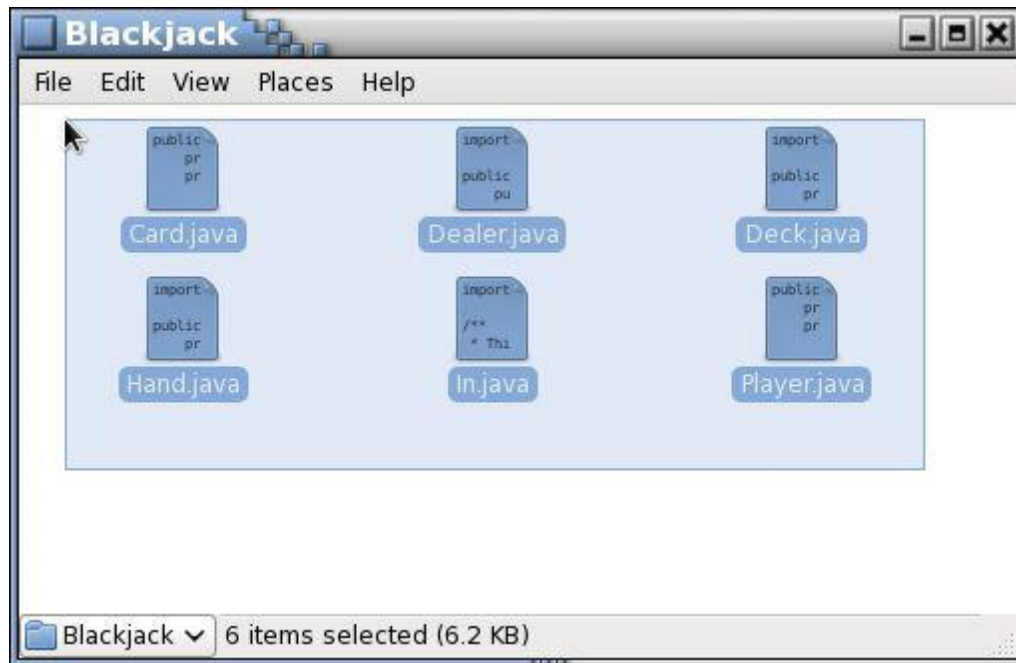
Drag the Blackjack folder from inside the ZIP file to the Desktop



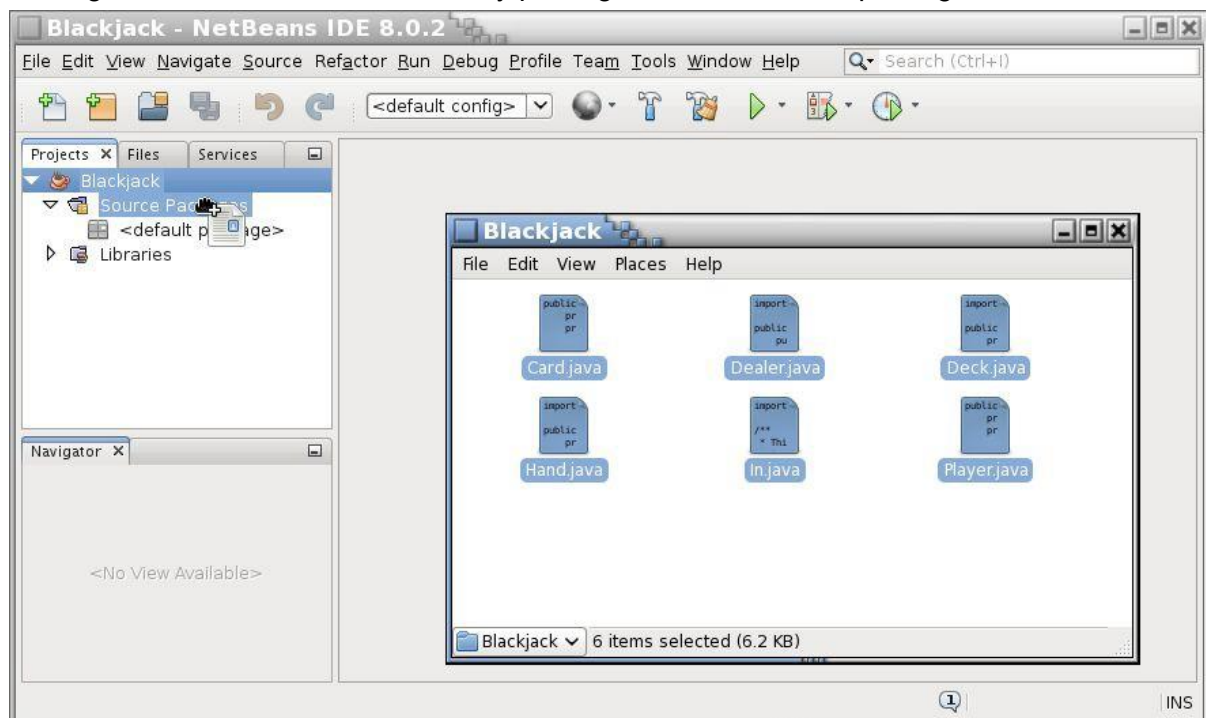
Double click on the Blackjack folder on the Desktop



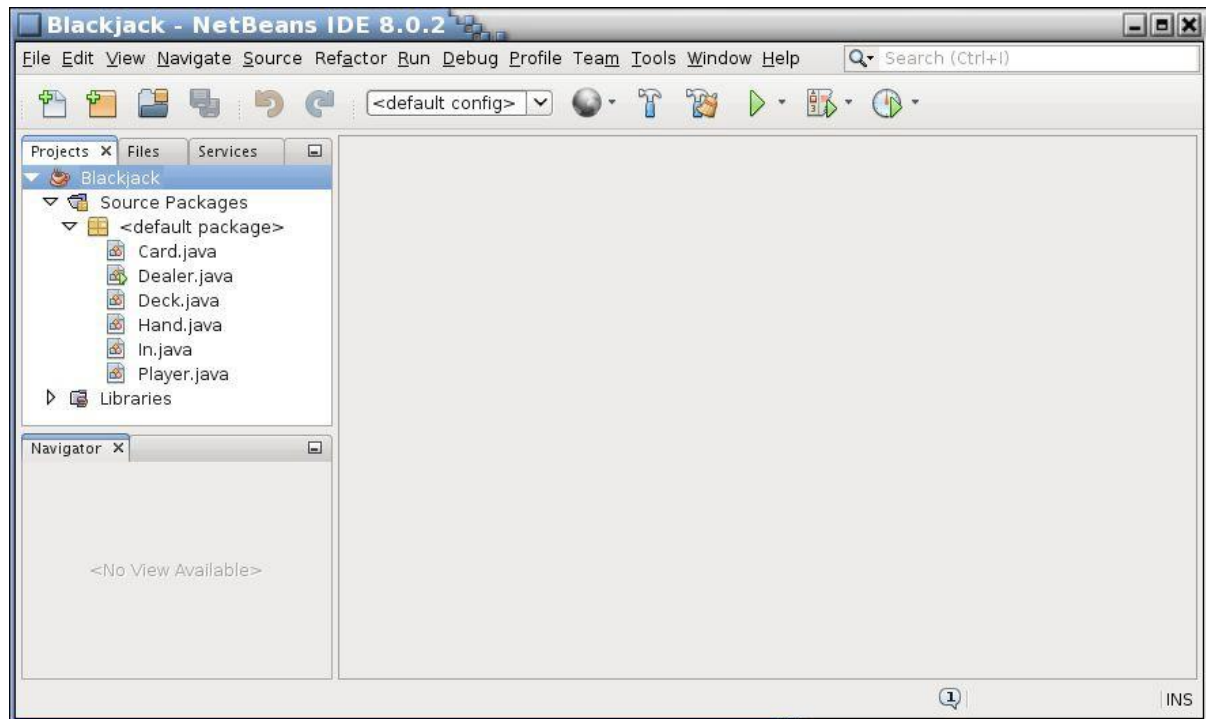
Select all Java files in the folder



Drag the Java files into the "Source Packages" folder inside your NetBeans project.
NOTE: If you're using MacOS, this doesn't work. Instead, copy the files with ⌘C (Command-C). Then paste them with ⌘V (Command-V). Try pasting them into "Source Packages" first. If that doesn't work, try pasting them into <default package>.



Expand to view the contents of the default package

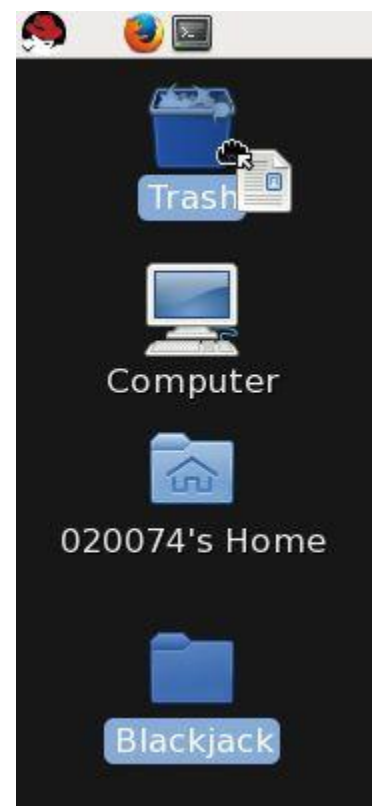


You may double click on any Java file to view the code for that class within NetBeans.

Double-click on the Player.java file to open the definition of the Player class.

Click on the green PLAY button to run your program. Select Dealer as the main class.

Before continuing, you may want to move the Blackjack folder from the Desktop to the Trash, since you have already copied the files into your NetBeans project.

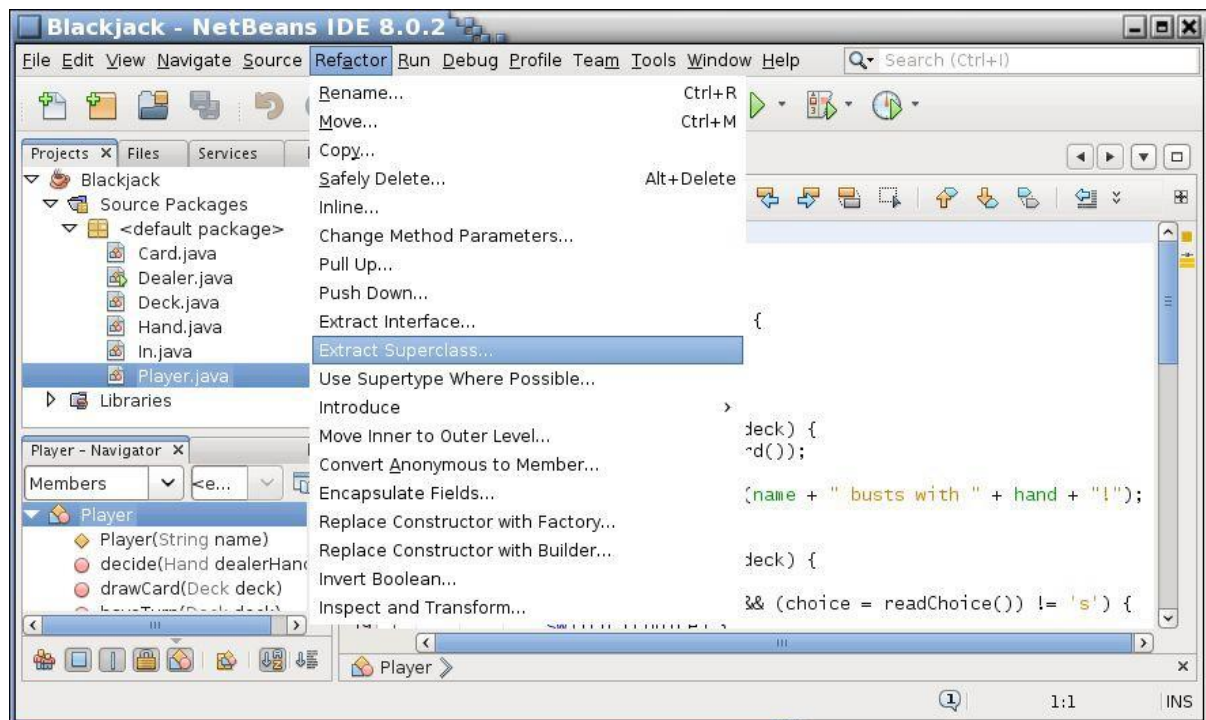


Refactoring code

Next, you will experiment with NetBeans' refactoring features. As shown in Study 6, it is possible to share common code between related classes by **inheriting** shared code from a superclass. NetBeans provides a refactoring tool to help you do this.

If you make a mistake, take a look at the final version of the lecture code (with the Person superclass). The code can be downloaded as a ZIP file from the top of the Study 7 module.

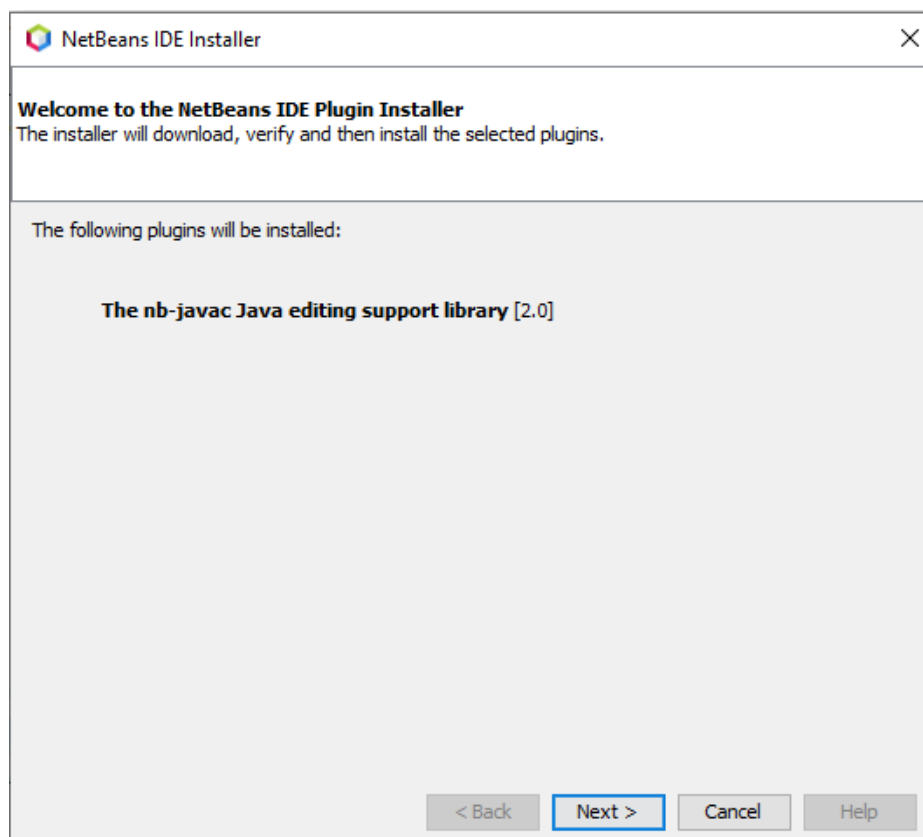
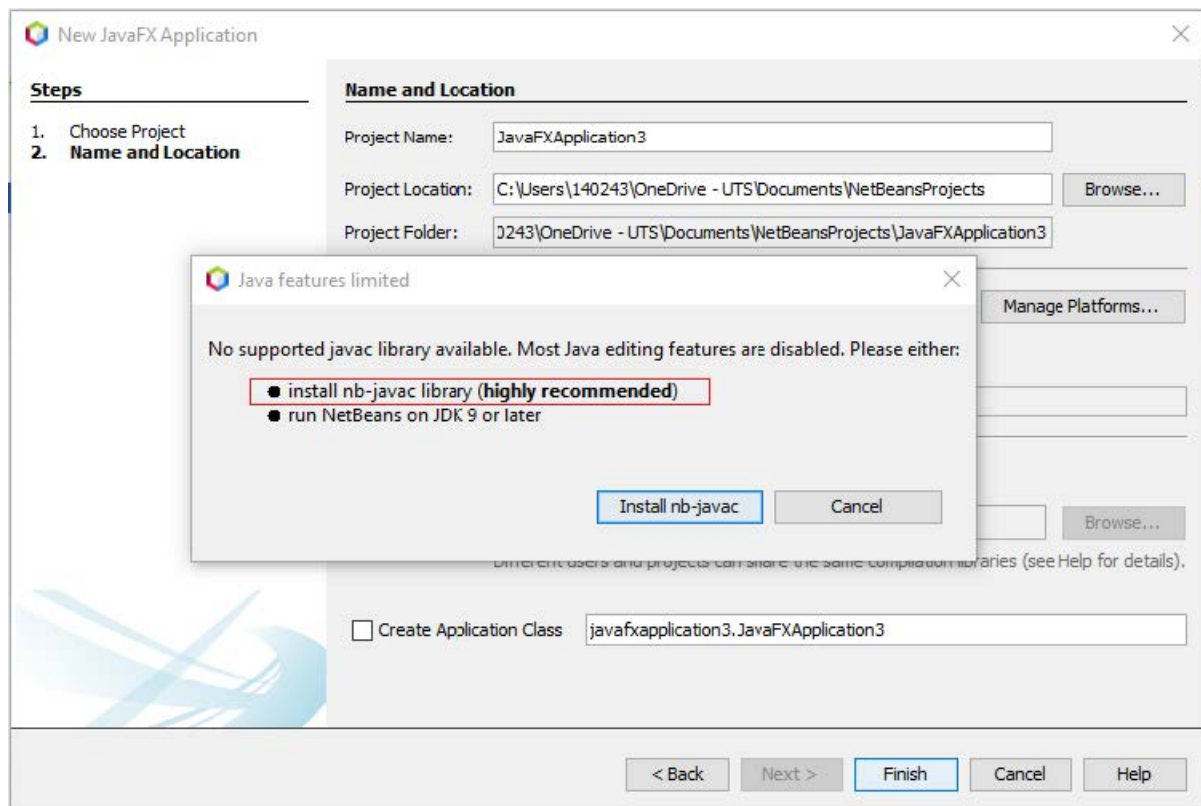
Open the Player.java file by double clicking on it. The Player class has a number of members that are in common with Dealer. We will extract those members into a superclass called Person. Select Extract Superclass... from the Refactor menu:



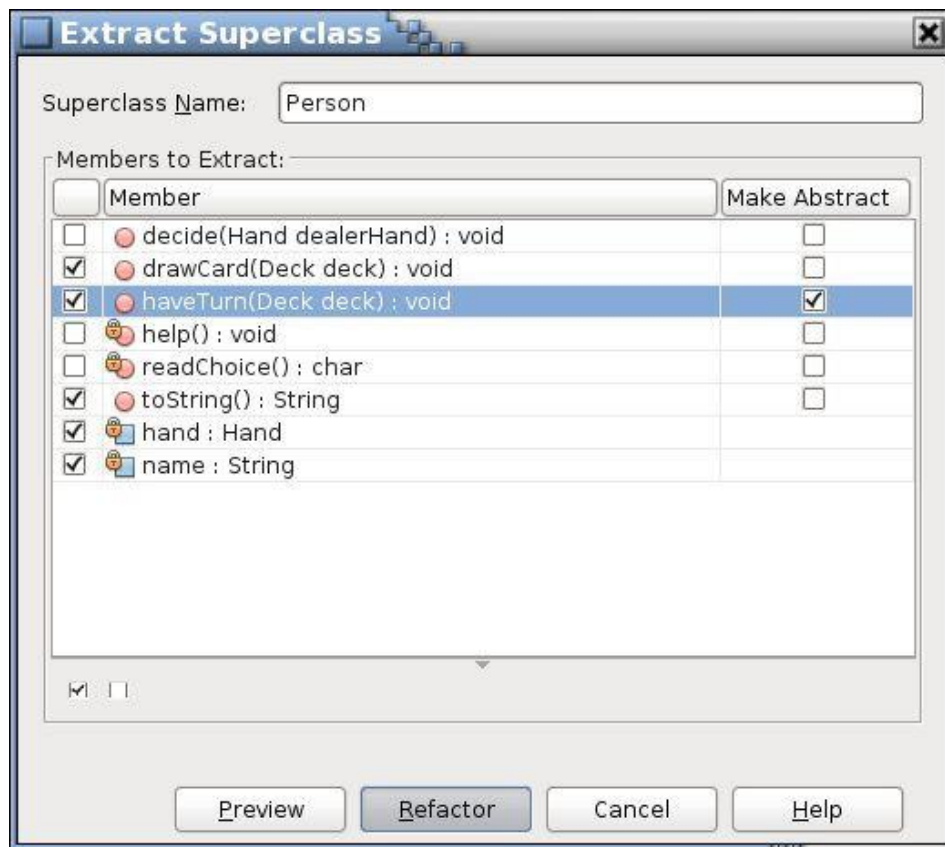
If you could not find the "Extract Superclass..." from the Refactor menu or it doesn't work, this is because the latest Java editing features haven't been involved in the default library in NetBeans. You need to download and install the "nb-javac Java editing support library" into your NetBeans. JavaFX project will also need it to activating most Java editing features.

If you are using all-in-one NetBeans(8.2) + JDK(8u111) bundle, you don't need to install any extension to activate Java editing features.

When Java features limited window pops up, please select "Install nb-javac" instead of upgrading JDK to later version.



Name the superclass Person. You need to select the checkboxes below from **both** the LEFT column and the RIGHT column. If you miss a checkbox, you'll need to start from scratch!



What this means is that we want the following members to be shared by both Player and Dealer:

- drawCard
- *haveTurn*
- toString
- hand
- name

From the RIGHT column, we must also select the “Make Abstract” checkbox for *haveTurn* because Player and Dealer have to implement this method differently.

Have a look at the resulting code. Notice some members have lifted from class Player up to class Person.

Note that the refactoring tool only made Player extend Person, but Dealer does not yet extend Person. You will need to do that one manually. Double-click on the Dealer.java file and modify the definition of class Dealer to extend Person. Delete from Dealer any members (both fields and methods) that are inherited. i.e. You can delete drawCard, toString, hand and name from Dealer since the Dealer class inherits these. The dealer **doesn't** inherit

haveTurn because the superclass didn't provide any implementation for it. So the Dealer has to keep its own implementation of *haveTurn*. You should add an `@Override` annotation above the Dealer's *haveTurn* method to indicate that it is defining its own version of this method. NetBeans might not have inserted an `@Override` annotation above the Player's *haveTurn* method either, so it would be good to add the annotation there, too.

Optional: If you wish, you may improve it further by defining a constructor in the superclass *Person* to initialize the name and the Hand. Subclasses can invoke this by calling `super` as the first line of their constructors. The lecture demo ran out of time to demonstrate the **super** keyword, but you can see an example of it in the Lecture code ZIP file.