Assignment 1

CompSci 230 S2 2014

Clark Thomborson

Submission deadline: 4pm, Friday 8 August 2014

Version 1.0 of 28 July 2014: based on hbv7.1.jar

Total marks on this assignment: 40. This assignment counts 4% of total marks in COMPSCI 230.

Note: if you started to work on this assignment when it was in V0.9, your development will be based on hbv7.jar. That's fine, please continue to work with this version. However hbv7 behaves differently to hbv7.1, so please be sure to indicate in your answer to Question 1 that you are observing hbv7.

Learning goals. By completing this assignment, you'll get some practical experience with exploring an OO design space. You'll also get some practical experience with using Eclipse, the de-facto industry standard IDE, for Java development.

Asking for help. If you become confused by Java syntax or semantics, or if you have difficulty using your Java development environment, you should look on the internet or ask anyone for help. If you do not understand the underlying concepts, you should start by reviewing the assigned readings and lecture slides. You are welcome to seek assistance from others on "learning the concepts" *after* you have done your assigned readings for this course.

Working independently. You are *not* allowed to have assistance from any other person when you are answering questions in this assignment, nor when you are designing and writing your class diagrams and code for this assignment. This must be your own work, done independently. You may gain design ideas or examples from the internet or a textbook. Anyone who gives you debugging assistance should **not** tell you "how to fix your bug"; however they might help you learn how to import code into your development environment, or how to use its debugger.

English grammar and spelling. You will not be marked down for grammatical or spelling errors in your submission. However if your meaning is not readily apparent to the marker, then you will lose some marks.

Resource requirements. You'll need

- a development environment for Java SE 7; you're encouraged to use Eclipse.
- Java 7 API documentation, in particular http://docs.oracle.com/javase/7/docs/api/java/awt/Point.html.
- a copy of the HuckleBuckle V7.1 codebase – this is available for download at https://www.cs.auckland.ac.nz/courses/compsci230s2c/assignments/hbv7.1.jar.

Submission instructions. You must submit electronically, using the Assignment Drop Box (https://adb.auckland.ac.nz/). Your submission will be **one document** (in PDF, docx, or odt) with your written answers to questions 1-6 below.

> Note. If you handwrite your answers, you must scan it before submission. The printer/photocopiers in some computer labs can scan documents and email them to your aucklanduni account. You should learn how to use this functionality well in advance of the submission deadline, to avoid last-hour frustrations and a penalty for a late submission.

Part 1: Importing a jarfile, then reverse-engineering its functionality

1) (**5 marks**) Import v7.1 of the HuckleBuckle codebase (`hbv7.1.jar`) into Eclipse, then run it with no command-line arguments. If your import is successful, a series of lines should be printed to the system.out console, starting with

```
        Playing HuckleBuckle on a 3 by 3 grid...
```

If your import is unsuccessful, or if you're confused by Eclipse, you may seek assistance from any source – ideally this would be a classmate in COMPSCI 230, so that you're "learning together". At the end of this document, you'll find some screenshots from my workstation, indicating the steps I took to import hbv7 into my build of Eclipse – these may help you figure out how to do this import into your own Eclipse environment. As soon as your import is successful, re-read the *Working independently* paragraph on the previous page. I'm happy for you to get assistance in figuring out how to use Eclipse, but you must make your own observations and interpretations of how HuckleBuckle behaves, so you should start working on your own at this point.

Run HuckleBuckle v7.1 five times, paying attention to how many times Sally moves before Harry calls out "Huckle Buckle Beanstalk!". You're playing the role of the Watcher in the use-case diagram in Figure 1 below. Discover the rules of this game by examining the output of this program, by reading its source code, and by reading the Wikipedia article (*http://en.wikipedia.org/wiki/Huckle_buckle_beanstalk*) on this general class of games.

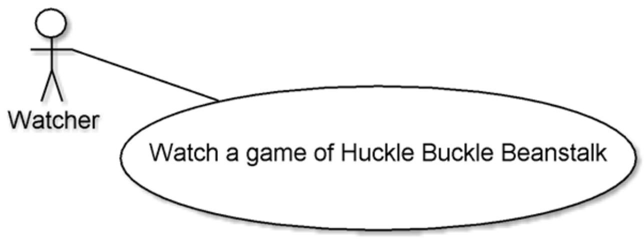


Figure 1. Use case diagram for `hbv7.1.jar with no commandline arguments.`

Submit: A written answer to this question, consisting of a few English sentences and a sample output from your runs of hbv7.1, describing (in general terms) what a Watcher will see whenever they run this application with no command-line arguments. To receive full marks, your submission must

[3 marks] use the following words: Harry, Sally, grid;

[1 mark] describe how Harry picks a place to hide an object; and

[1 mark] describe how long it took Sally to find the hidden object, in each of your five observations. Please be sure to indicate, in your answer, whether you're using hbv7.jar or hbv7.1.jar.

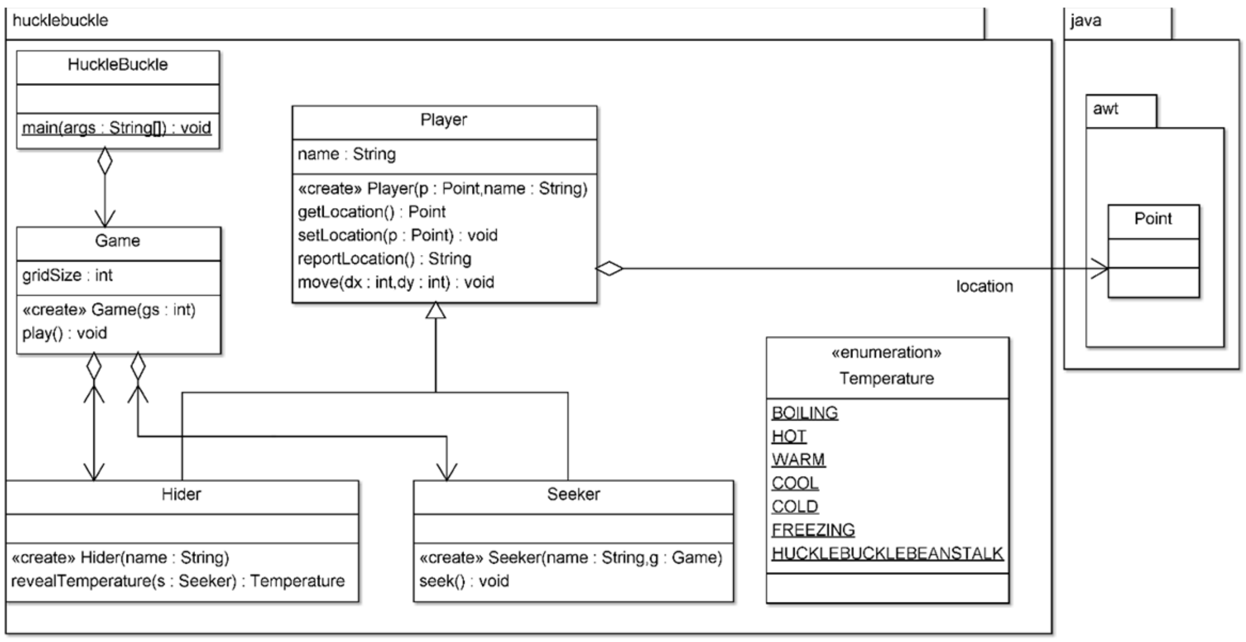
Part 2: Exploring the design space of a simple Java program

I made many decisions when developing the class structure and implementation of hbv7.jar. You'll find a UML diagram of my class structure in Figure 2 on the next page. You can see my implementation in the java source code. Some of my decisions are questionable – if I were to "clean up" this code, I'd reconsider these decisions. Most of my decisions were arbitrary or unimportant – I chose one of many reasonably-good ways to get the job done. In this part of the assignment, you'll be exploring the design space of this codebase, as you create several new versions of the codebase by adding a feature to my v7 or v7.1 codebase. You're not expected to have a strong sense of Java style, but you are expected to be learning about OO design decisions which make it easier – or harder – for a developer to extend an existing design with correct and easily-understandable code.

2) (**5 marks**) Start a new Eclipse project called hbv8, then add all sourcefiles from hbv7.1 to this project. Note: you may use hbv7.jar for this question. Now modify the coding of `main()`, so that it accepts an optional second argument: the name of the Seeker. You'll also have to modify the coding of Game, so that it passes this argument (if supplied) instead of "Sally" to the Seeker constructor. At the end of this handout, you'll see a screenshot indicating how you can enter command-line arguments in the Eclipse environment by typing in the "Program Arguments" area of the Run dialog.

Submit: a written answer to this question, consisting of a listing of your modified main() method, and a listing of your modified constructor for Game. Note: you can cut-and-paste from an edit-window in your Eclipse frame into your written submission, but you may have to clear the highlighting on this text so that it displays properly – you may ask anyone for assistance with this formatting step.

Your modified main() method must accept an optional second argument [**1 mark**], it must transmit this argument to the Game constructor [**1 mark**], and it must print the message "`Usage: hbv8 [gridSize [yourName]]`" if more than two commandline arguments are provided [**1 mark**]. Your modified Game constructor must pass the string "Sally" to the Seeker constructor if the second commandline argument is not provided [**1 mark**], otherwise it must pass the second commandline argument to the Seeker constructor [**1 mark**].



hucklebuckle
java
awt
Point
HuckleBuckle
main(args : String[]) : void
Player
name : String
«create» Player(p : Point,name : String)
getLocation() : Point
setLocation(p : Point) : void
reportLocation() : String
move(dx : int,dy : int) : void
location
Game
gridSize : int
«create» Game(gs : int)
play() : void
«enumeration»
Temperature
BOILING
HOT
WARM
COOL
COLD
FREEZING
HUCKLEBUCKLEBEANSTALK
Hider
«create» Hider(name : String)
revealTemperature(s : Seeker) : Temperature
Seeker
«create» Seeker(name : String,g : Game)
seek() : void

Figure 2. Class diagram of hbv7.1.jar.

3) (**10 marks**) Add a new class called `HumanSeeker` to your `hbv8`. The `seek()` method of your new class should accept your console input – you'll be controlling the Seeker instance, whenever you supply a second commandline argument. At the end of this handout, you'll see a screenshot of my hbv8, indicating the input and output conventions: you'll be typing single characters (n, s, e, or w) to make the Seeker move North, South, East, or West (respectively); and if you type a "q" then the program should terminate with the message ("I give up. I can't find you!"). You have several design options, including the following: 1) the HumanSeeker could inherit from Seeker; 2) the HumanSeeker could inherit from Player; 3) the Seeker could be an abstract class with two subclasses, HumanSeeker and ComputerSeeker. In your hbv8, you should use the first of these options, and you'll find some coding ideas in the screenshot at the end of this handout.

Submit: A written answer to this question, consisting of a listing of your HumanSeeker class [**8 marks**], and a brief English description [**2 marks**] of how you modified a method in Game (or in Seeker, or in some other class) so that a HumanSeeker is instantiated whenever a second commandline argument is presented to your hbv8.

4) (**8 marks**) Create a new project called hbv9, and copy your v8 source code into this project. Now modify this code to implement the second design option listed in question 3: HumanSeeker should inherit from Player, rather than from Seeker.

   Submit: A written answer to this question, consisting of a listing of your HumanSeeker class [**6 marks**], and a brief English discussion [**2 marks**] of any difficulties you encountered while implementing this design.

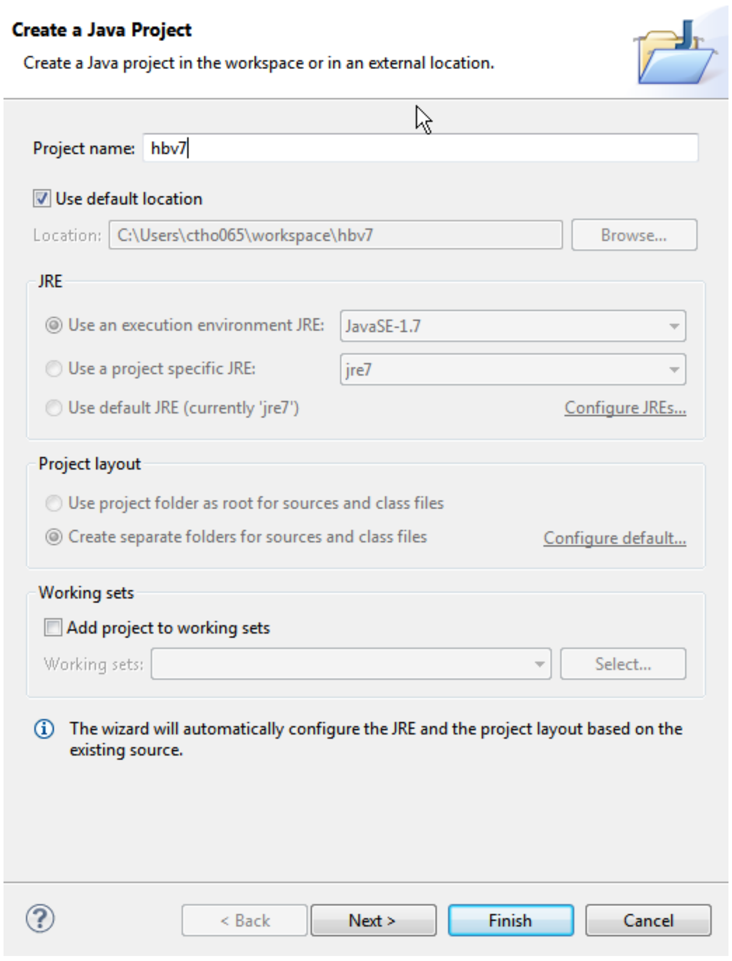
5) (**8 marks**) Create a new project called hbv10, and copy your v8 source code into this project. Now modify this code to implement the third design option listed in question 3: Seeker should be an abstract class with two subclasses, HumanSeeker and ComputerSeeker. The ComputerSeeker should have the same behaviour as the Seeker in hbv7.1. Note: if you based your development on hbv7.jar, your ComputerSeeker should have the same (buggy ;-) behaviour as my v7 seek().

   Submit: A written answer to this question, consisting of a listing of your HumanSeeker class [**6 marks**], and a brief English discussion [**2 marks**] of any difficulties you encountered while implementing this design.

6) (**4 marks**) Based on your experience with these three implementations, do you see any advantage in the third option (hbv10) which might make it the best choice for this program?

   Submit: A written answer to this question, consisting of a few English sentences.

Appendix: Screenshots



Create a Java Project
Create a Java project in the workspace or in an external location.

Project name: hbv7

☑ Use default location

Location: C:\Users\ctho065\workspace\hbv7

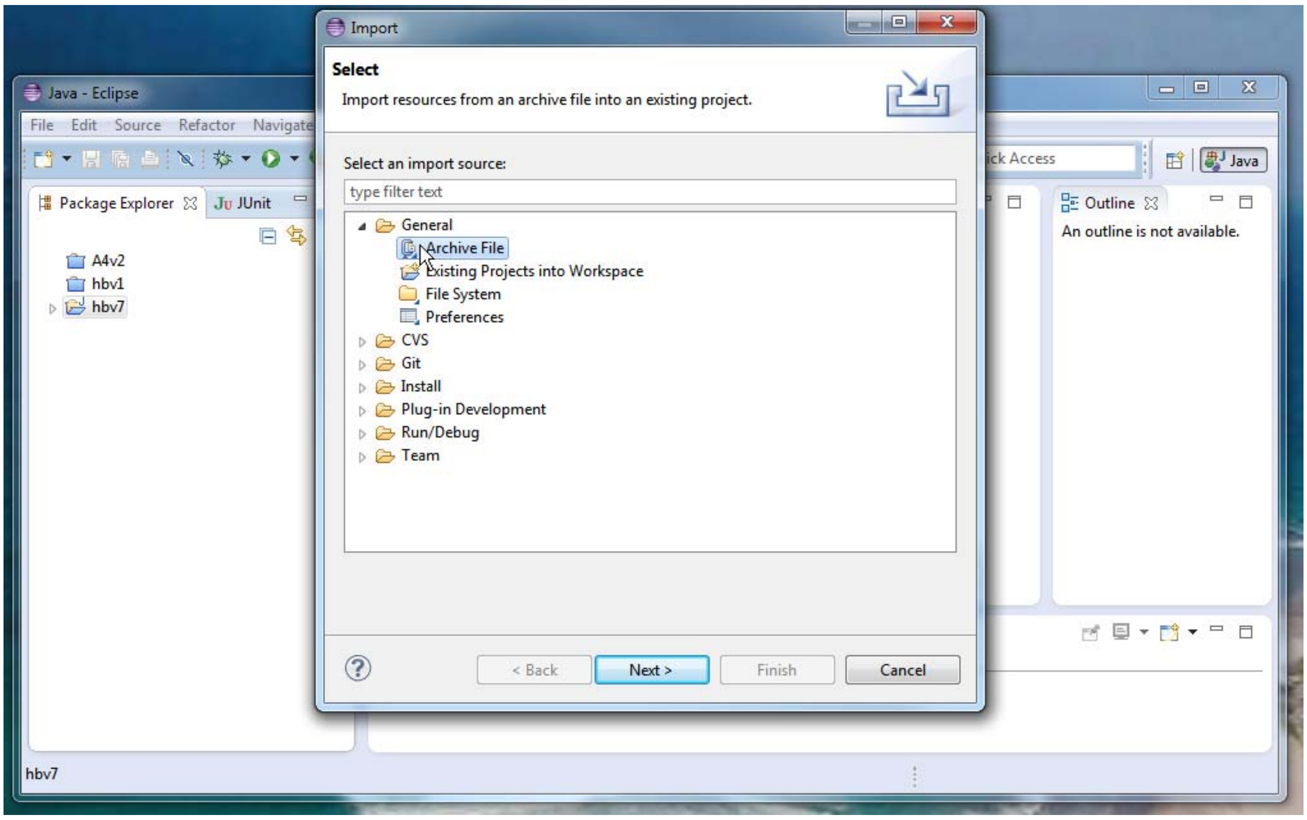
JRE
- ◉ Use an execution environment JRE: JavaSE-1.7
- ○ Use a project specific JRE: jre7
- ○ Use default JRE (currently 'jre7')

Project layout
- ○ Use project folder as root for sources and class files
- ◉ Create separate folders for sources and class files

Working sets
- ☐ Add project to working sets

Working sets:

The wizard will automatically configure the JRE and the project layout based on the existing source.



Java - Eclipse

Import

Select

Import resources from an archive file into an existing project.

Select an import source:

type filter text

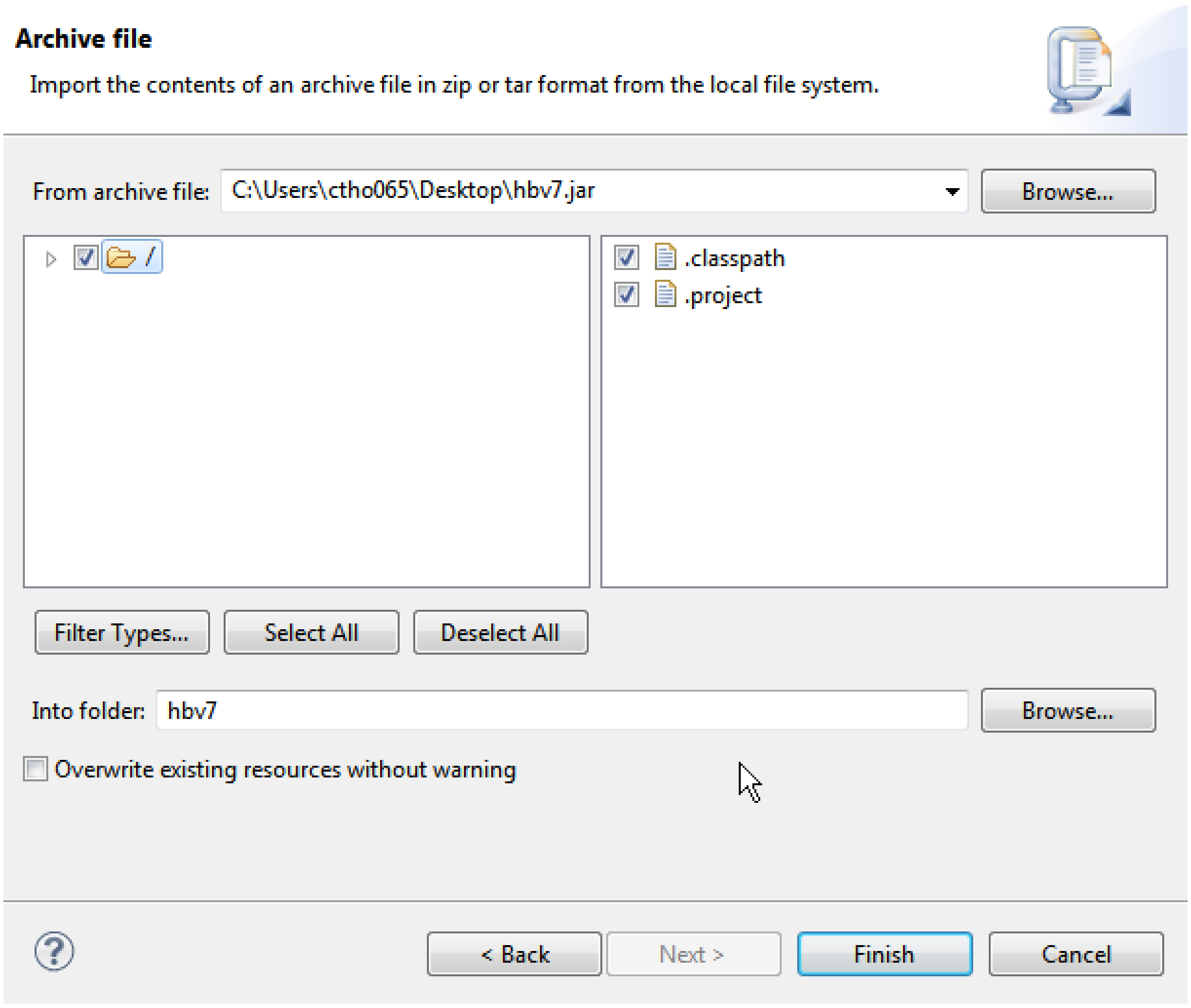
- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- CVS
- Git
- Install
- Plug-in Development
- Run/Debug
- Team

< Back | Next > | Finish | Cancel

Package Explorer: A4v2, hbv1, hbv7

An outline is not available.

hbv7



Archive file

Import the contents of an archive file in zip or tar format from the local file system.

From archive file: C:\Users\ctho065\Desktop\hbv7.jar

[✓] /

[✓] .classpath

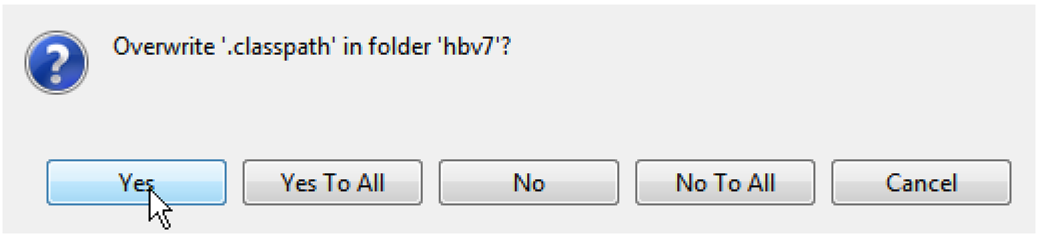
[✓] .project

Filter Types... | Select All | Deselect All

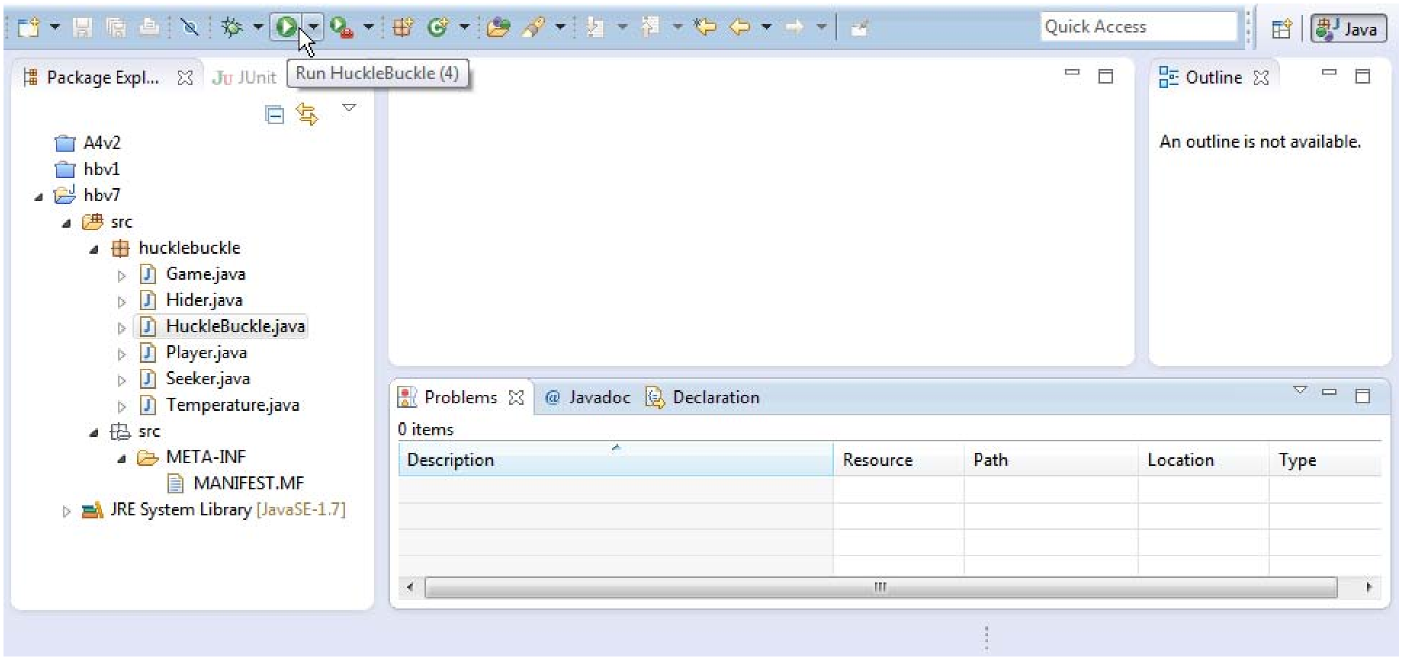
Into folder: hbv7

[ ] Overwrite existing resources without warning

< Back | Next > | Finish | Cancel



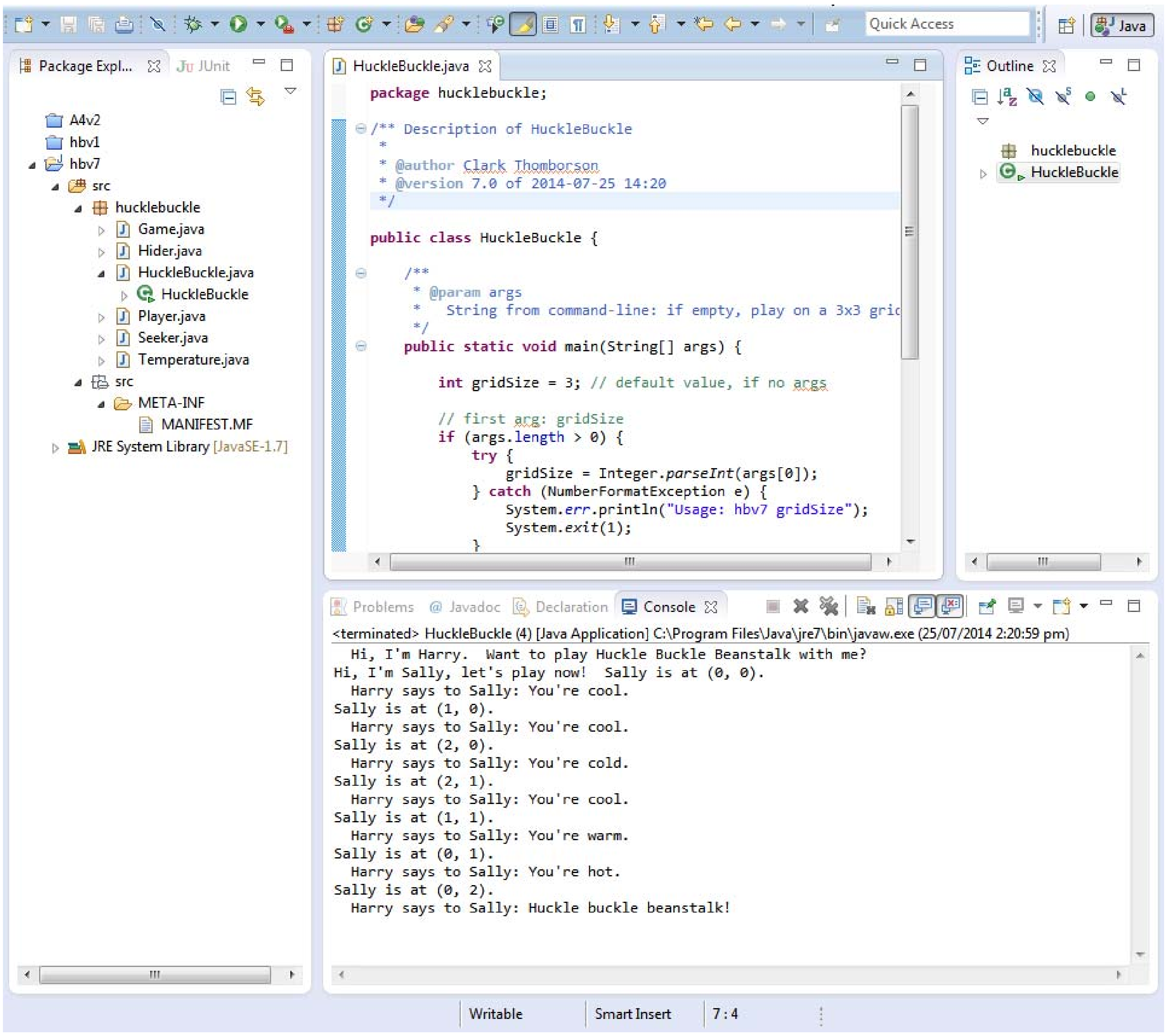
Overwrite '.classpath' in folder 'hbv7'?



An outline is not available.

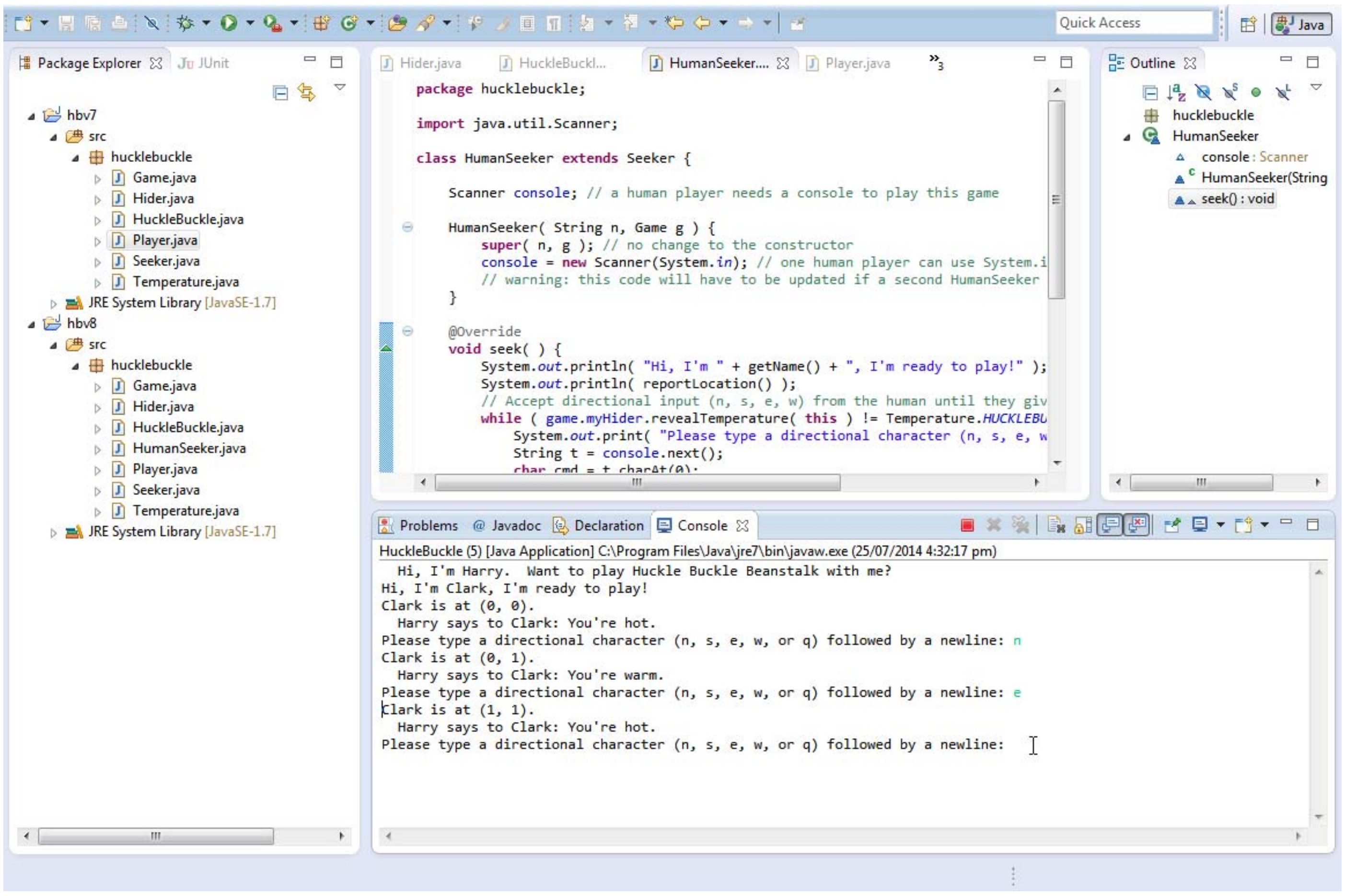
0 items

Description	Resource	Path	Location	Type



```java
package hucklebuckle;

/** Description of HuckleBuckle
 *
 * @author Clark Thomborson
 * @version 7.0 of 2014-07-25 14:20
 */

public class HuckleBuckle {

    /**
     * @param args
     *   String from command-line: if empty, play on a 3x3 gri
     */
    public static void main(String[] args) {

        int gridSize = 3; // default value, if no args

        // first arg: gridSize
        if (args.length > 0) {
            try {
                gridSize = Integer.parseInt(args[0]);
            } catch (NumberFormatException e) {
                System.err.println("Usage: hbv7 gridSize");
                System.exit(1);
            }
```

```
<terminated> HuckleBuckle (4) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (25/07/2014 2:20:59 pm)
  Hi, I'm Harry.  Want to play Huckle Buckle Beanstalk with me?
Hi, I'm Sally, let's play now!  Sally is at (0, 0).
  Harry says to Sally: You're cool.
Sally is at (1, 0).
  Harry says to Sally: You're cool.
Sally is at (2, 0).
  Harry says to Sally: You're cold.
Sally is at (2, 1).
  Harry says to Sally: You're cool.
Sally is at (1, 1).
  Harry says to Sally: You're warm.
Sally is at (0, 1).
  Harry says to Sally: You're hot.
Sally is at (0, 2).
  Harry says to Sally: Huckle buckle beanstalk!
```



```
package hucklebuckle;

import java.util.Scanner;

class HumanSeeker extends Seeker {

    Scanner console; // a human player needs a console to play this game

    HumanSeeker( String n, Game g ) {
        super( n, g ); // no change to the constructor
        console = new Scanner(System.in); // one human player can use System.i
        // warning: this code will have to be updated if a second HumanSeeker
    }

    @Override
    void seek( ) {
        System.out.println( "Hi, I'm " + getName() + ", I'm ready to play!" );
        System.out.println( reportLocation() );
        // Accept directional input (n, s, e, w) from the human until they giv
        while ( game.myHider.revealTemperature( this ) != Temperature.HUCKLEBU
            System.out.print( "Please type a directional character (n, s, e, w
            String t = console.next();
            char cmd = t.charAt(0);
```

HuckleBuckle (5) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (25/07/2014 4:32:17 pm)

```
  Hi, I'm Harry.  Want to play Huckle Buckle Beanstalk with me?
Hi, I'm Clark, I'm ready to play!
Clark is at (0, 0).
  Harry says to Clark: You're hot.
Please type a directional character (n, s, e, w, or q) followed by a newline: n
Clark is at (0, 1).
  Harry says to Clark: You're warm.
Please type a directional character (n, s, e, w, or q) followed by a newline: e
Clark is at (1, 1).
  Harry says to Clark: You're hot.
Please type a directional character (n, s, e, w, or q) followed by a newline:
```