CS 143 Lab 1 Programming Specifications

Checkpoint Due 11:59PM Friday July 6
Lab Due 11:59PM Monday July 9

Problem Overview: This lab is intended to give you more practice with classes. It will also extend your previous work with single-dimensional arrays, into the realm of 2-dimensional arrays. The code you write in this lab will form the foundation for Lab 2.

The `YardDog` Class: You are going to write a `YardDog` class that simulates bone-burying dogs in yards. Yards are rectangular grids (regular, 2D arrays). There is a fence occupying the top and bottom rows of the yard array, as well as the leftmost and rightmost columns of the yard array. Three bones labeled 'A', 'B' and 'C' will be randomly buried in the yard within the fence line. The dog will dig holes within the fence line of the yard to locate a bone. A sample yard, with fence line and three bones, is shown at the right.

```
+------+
| B    |
|      |
|   A  |
|      |
| C    |
|      |
+------+
```

		x		
	x	B	x	
		x		

Finding Bones: The dogs dig such large holes that they locate a bone if they dig a hole exactly over a bone, immediately north, immediately south, directly east or directly west of where a bone is buried. This is depicted in the sample portion of the yard grid shown at the left, where bone B is found if the dog digs a hole exactly where the bone is buried, or in one of the other locations marked by x.

Stepwise Refinement: You will implement the `YardDog` class in 5 stages. Do not proceed to later stages until you have completed the earlier stages.

Stage 1—Lab 1 Checkpoint: Define and test a `YardDog` class that meets the specifications given in the table below. Review BJP Chapter 7 on arrays and Chapter 8 on classes as needed.

`YardDog` Class	
Class *Constants*	Description

`public static final:`
`int DEFAULT_ROWS = 15`
`int DEFAULT_COLS = 20`
`int MIN_ROWS = 5`
`int MIN_COLS = 5`
`int MAX_COLS = 75`
NOTE: These dimensions include the fence lines, so a yard with 15 rows has 13 rows in which to bury bones and dig holes.
	These class constants define the minimum, maximum and default dimensions of the 2D `yard` array.
	• Recall that `static` items belong to the entire class. There is only one such static item, regardless of how many *instances* (objects) of the class may exist.
	• Since they are *constants* (`final`), they cannot be changed once initialized, so they don't need the protection of `private` access. Clients can have direct access to them without going through accessor ("get") methods.

Instance Data Fields (non-`static`) Each `YardDog` object has its own data	Description
`private char yard[][]`	The regular 2D array that will hold the fence, the 3 bones, the symbols to represent holes that have been dug, etc. The dimensions of the yard can vary from `YardDog` object to `YardDog` object.
`private String name`	The name of the dog digging all the holes.

Do not declare any other data fields than those specified above. All other variables should be declared locally in the methods that need them.

`YardDog` Class (continued)

Constructor Method	Description

`public YardDog(int numRows, int numCols, String name)`
 	
WARNING: When using the `new` operator inside the `YardDog` constructor, **do not redeclare the `yard` array field** of the `YardDog` class! The example shown in BJP 7.5 is as follows:
`double[][] temps = new double[3][5];`
This Java statement both *declares* and *constructs* the 2D `temps` array. Those two operations could have been separated as follows:
`double[][] temps; //declare`
`temps = new double[3][5];//construct`
	3-argument *constructor* creates a `YardDog` object as follows:
	• `throw` an `IllegalArgumentException` if the `numRows` is less than `MIN_ROWS`, or `numCols` is less `MIN_COLS` or greater than `MAX_COLS`.
	
	REMINDER: Section 4.4 of the BJP textbook showed how to throw exceptions.
	• Use the `new` operator to construct the 2D `yard` array object having the number of rows and columns specified by the corresponding parameter values. ←See the WARNING at the left.
	• Call the `buildYard()` method described below.
	• Set the `name` field to the value of the `name` parameter. Use keyword `this` as discussed in BJP 8.3 to "unshadow" the `name` field shadowed by the `name` parameter.

Mutator Methods	Description

`public void buildYard()`
`+-------------+`
`|  B  |`
`|  A  |`
`|  C  |`
`+-------------+`
	For the Stage-1 checkpoint, this mutator method should simply fill the `yard` with periods (`'.'`) as shown in the `print()` method described below.
	In Stage 3 of this lab, this method will fill the 2D `yard` in the manner described in the overview at the top of this document. An example of a possible Stage-3 `yard` is shown at the left.

`public void setElement(int row, int col, char ch)`	This mutator method should `throw` an `IllegalArgumentException` if either `row` or `col` parameter value is outside the boundaries of the `yard` array. Otherwise, this method assigns `yard[row][col]` the value of `ch`.

`YardDog` Class (continued)	
Accessor Methods	Description
`public void print()` `.....` `.....` `.....` `.....` `.....` `.....` `.....` `.....` `.....` `.....` **`Fido`**	This method displays the contents of the 2D `yard` field, followed by the name of the dog. In Stage 1, the constructor method for the Lab 1 checkpoint fills the 2D `yard` fiel`d with periods (`'.'`) Sample output for a Stage-1 `YardDog` object having a `yard` with dimensions 10 rows x 20 columns, and a dog named Fido, is shown at the left. **HINT:** See the nested `for`-loop code in the *Regular Two-Dimensional Arrays* section of BJP 7.5 for an example of how to display each element of a 2-D array. **NOTE:** We do not want to display a space character (`' '`) after each of the `yard` characters as shown in the book's example. If we did that, a yard containing the maximum 75 columns could not be displayed properly in the console window.
`public String getName()`	This accessor method returns value of the `name` field.
`public int getNumRows()`	This accessor method returns the number of rows in the 2D `yard` array. **HINT:** Section 7.5 in the BJP textbook shows the number of rows is the `length` of the array
`public int getNumColumns()`	This accessor method returns the number of columns in the 2D `yard` array. **HINT:** Section 7.5 in the BJP textbook shows the number of columns is equal to the length of the row. Since all rows are the same length in the 2D `yard` array, the number of columns is the same as the length of row 0.
`public char elementAt(int row,` `int column)`	This accessor method should `throw` an `IllegalArgumentException` if either parameter value is outside the boundaries of the `yard` array. Otherwise, this method returns the value of the character at the specified `row` and `column` position of the `yard`. **WARNING:** A 2D array with 5 rows and 5 columns has rows and columns numbered 0 through 4, NOT 1 through 5.

Test Stage-1 `YardDog` class: The `YardDogClient.java` file posted with this assignment contains code for testing each stage of this lab. The bodies of the tester methods are commented out as they won't compile correctly until you have implemented the methods that they test. Recall the following about "server" and "client" classes:

- Files such as `YardDog.java` that define server classes can be **compiled only**—they cannot be run.
    - Use the "green plus" icon in the jGRASP toolbar to compile the file, instead of the "red running man icon" that both compiles and runs the file.
- The `.java` server and client files should be in the same folder.

When you uncomment the body of the `stage1N3_Test()` method in the `YardDogClient.java` file, the following I/O session can be generated, with user input shown boldfaced and underlined here, for easy identification only:

```
Stages of YardDog testing
A. Test Stage 1 or Stage 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? A

DEFAULT_ROWS = 15, DEFAULT_COLS = 20,
MIN_ROWS = 5, MIN_COLS = 5, MAX_COLS = 75
YardDog fido = new YardDog(10,10,"Fido")
Dog named Fido has yard with 10 rows and 10 columns
..........
..........
..........
..........
..........
..........
..........
..........
..........
..........
Fido
Enter a row and column number (negative values to quit) 0 0
fido.elementAt(0,0) is '.'
After fido.setElement(0,0,'#'):
#.........
..........
..........
..........
..........
..........
..........
..........
..........
..........
Fido
Enter a row and column number (negative values to quit) 5 5
fido.elementAt(5,5) is '.'
After fido.setElement(5,5,'#'):
#.........
..........
..........
..........
..........
.....#....
..........
..........
..........
..........
Fido
Enter a row and column number (negative values to quit) 10 10
Row 10 Column 10 caused exception.
Enter a row and column number (negative values to quit) -1 -1
Row -1 Column -1 caused exception.
Stages of YardDog testing
```

```
A. Test Stage 1 or Stage 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? e

Good-bye!
```

Be sure to read the *How I Grade Checkpoints* page in the Quick Links module on the Canvas website for the class.

End Stage 1 Lab 1 Checkpoint

Stage 2 Specifications: All classes that define objects should contain a *default constructor* (no arguments) and an appropriate `toString()` method. You will add these methods to your `YardDog` class, and then test them using the uncommented `Stage2_Test()` method in the `YardDogClient.java` file.

- `public YardDog()` – This default constructor should use the keyword `this` appropriately to call the 3-argument constructor you defined in Stage 1. This technique was discussed near the end of BJP 8.3. The resulting `YardDog` object should have `DEFAULT_ROWS`, `DEFAULT_COLS`, and dog's `name` "Dog".
- `public String toString()` – This method returns the text representation of the `YardDog` object.
    - For example, if the client program had declared a `YardDog` object named `dog`, with a yard of 10 rows x 20 columns and the dog's name was Fido, a call to `System.out.print(dog)`, or `System.out.print(dog.toString())`, should display an image such as that shown for the `print()` method in Stage 1.
    - **This method does not display anything**—it *returns* a `String` object that can be displayed.
    - The `toString()` method declares, creates and returns a local `String` object that is built by using repeated concatenation operations to concatenate the various characters that comprise the `yard`, plus the name of the dog at the end. Be sure to concatenate a `'\n'` character to the end of each row of characters, and following the dog's name.

A sample I/O session is shown below. Again, user input is shown boldfaced and underlined only for easy identification.

```
Stages of YardDog testing
A. Test Stage 1 or Stage 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? B

YardDog dog = new YardDog()
YardDog named Dog has yard with 15 rows and 20 columns
System.out.print(dog) shows:
....................
....................
....................
....................
```

```
..................
..................
..................
..................
..................
..................
..................
..................
..................
..................
..................
Dog
Stages of YardDog testing
A. Test Stage 1 or Stage 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? E

Good-bye!
```

Stage 3 Specifications: Replace the Stage-1 version of the `buildYard()` method with one that performs the following tasks:

- Fill the `yard` with spaces (`' '`)
- Build a fence around the perimeter of the yard using `'+'`, `'-'`, and `'|'` characters as shown in the sample I/O session below.
- Bury bones `'A'`, `'B'` and `'C'` in random locations in the `yard`.
    - Generate random row and column coordinates for the bones through appropriate use of the `nextInt()` method of the Java `Random` object introduced in BJP 5.1.
    - The bones must be buried within the fence line. That is, they may not be buried in the top or bottom rows of the 2D `yard`, nor in the leftmost or rightmost columns of the 2D `yard`.
    - All locations within the yard's fence line should have an equal probability of holding a buried bone.
    - No two bones may be buried in the same hole.
    - Did you know you can create a `for`-loop such as the following? `for(char bone='A'; bone<='C'; bone++)...`
- You may have the `buildYard()` method call `private` instance methods (non-static) of your own design, if the `buildYard()` method is getting too long or confusing. For example, you might want separate `buildFence()` or `buryBones()` methods for `buildYard()` to call.

You may test your `buildYard()` method using `YardDogClient.java` as shown below. Keep in mind that locations of bones are randomly generated, so your actual I/O session will vary!

```
Stages of YardDog testing
A. Test Stage 1 or Stage 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? A
```

```
DEFAULT_ROWS = 15, DEFAULT_COLS = 20,
MIN_ROWS = 5, MIN_COLS = 5, MAX_COLS = 75
YardDog fido = new YardDog(10,10,"Fido")
Dog named Fido has yard with 10 rows and 10 columns
+--------+
|    B   |
|      C |
|        |
|        |
|        |
|        |
|  A     |
|        |
+--------+
Fido
Enter a row and column number (negative values to quit) 1 5
fido.elementAt(1,5) is 'B'
After fido.setElement(1,5,'#'):
+--------+
|    #   |
|      C |
|        |
|        |
|        |
|        |
|  A     |
|        |
+--------+
Fido
Enter a row and column number (negative values to quit) 2 7
fido.elementAt(2,7) is 'C'
After fido.setElement(2,7,'#'):
+--------+
|    #   |
|      # |
|        |
|        |
|        |
|        |
|  A     |
|        |
+--------+
Fido
Enter a row and column number (negative values to quit) -1 -1
Row -1 Column -1 caused exception.
Stages of YardDog testing
A. Test Stage 1 or Stage 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? E

Good-bye!
```

Stage 4 Specifications: Implement a `boneFound()` instance method for the `YardDog` class.

- The method should have the following header:

  ```
  public boolean boneFound(int row, int col)
  ```

- This method returns `true` if the element at location `row`, `col` of the yard "finds" a bone as described in the overview on the first page of this document. Otherwise, it returns `false`.
  - If the `row` or `col` parameter does not provide a valid coordinate within the fence line of the `yard` array, return `false`—do not throw an exception.
- This method does not display anything.

Uncomment the body of the `stage4_Test()` method in the `YardDogClient.java` file to generate an I/O session such as the following:

```
Stages of YardDog testing
A. Test Stage 1 and 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? c

+------------------+
|                  |
|           A      |
|                  |
|                  |
|                  |
|                  |
|        C         |
|                  |
|                  |
|                  |
| B                |
|                  |
|                  |
+------------------+
Dog
x will be displayed at all locations that "find" a bone.
+------------------+
|           x      |
|          xxx     |
|           x      |
|                  |
|                  |
|        x         |
|       xxx        |
|        x         |
|                  |
| x                |
|xxx               |
| x                |
|                  |
+------------------+
Dog
```

```
Stages of YardDog testing
A. Test Stage 1 and 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? e

Good-bye!
```

Stage 5 Specifications: Implement a `digHoles()` instance method for the `YardDog` class. The `digHoles()` method will continue to dig holes until a bone is "found". It then returns the number of holes that were dug until a bone was located.

- `public int digHoles()` is the method header
- The `digHoles()` method performs the following tasks:
  - Display the `YardDog` by calling an appropriate `YardDog` method.
  - Ask the user for row and column coordinates in which to dig the first hole.
  - Initialize a hole counter to 1.
  - *As long as* the hole just dug does not find a bone. . .
    - Place `'.'` in the yard at the row/column coordinates last given by the user.
    - Display the `YardDog` by calling an appropriate `YardDog` method.
    - Ask the user for row and column coordinates in which to dig the next hole.
    - Increment the hole counter.
  - Place `'H'` in the yard at the row/column coordinates last given by the user.
  - Display the `YardDog` by calling an appropriate `YardDog` method.
  - Return the final hole count.

NOTE: If the user enters invalid coordinates for the hole, the program will terminate. Without knowing more than we know now, this is correct behavior.

Test the `digHoles()` method by uncommenting the `stage5_Test()` method in the `YardDogClient.java` file. A sample I/O session should look like the following:

```
Stages of YardDog testing
A. Test Stage 1 and 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? D

+--------+
|        |
|        |
|        |
|  B     |
|        |
|        |
|        |
|   C    |
|     A  |
+--------+
Digger
Enter row and column for hole to dig: 7 2
```

```
+--------+
|        |
|        |
|        |
|  B     |
|        |
|        |
| . C    |
|     A  |
+--------+
Digger
Enter row and column for hole to dig: 7 7
+--------+
|        |
|        |
|        |
|  B     |
|        |
|        |
| . C  . |
|     A  |
+--------+
Digger
Enter row and column for hole to dig: 8 4
+--------+
|        |
|        |
|        |
|  B     |
|        |
|        |
| . C  . |
|   H A  |
+--------+
Digger
Digger dug 3 holes to locate a bone.
Stages of YardDog testing
A. Test Stage 1 and 3 code
B. Test Stage 2-default constructor; toString()
C. Test Stage 4-boneFound())
D. Test Stage 5-digHoles()
E. Quit testing
Your menu choice(A-E)? E

Good-bye!
```

Style and Other Grading Specifications: All programs in this class must adhere to the items listed in the *Good Style Specifications* document posted in the Quick Links module on the Canvas website for this class.

Be sure read the *How I Grade Weekly Programming Lab Projects* page also posted in the Quick Links module.