

# WES-CS GROUP MEETING #13

## Exercise 1: Programming Connect-4

For this exercise we will work on a program that allows two people to play a game of Connect-4 against each other, using your program instead of the actual Connect-4 game.

To see how the program should work, start by playing the version written by WES-CS Team Leader Jesse Benson.

After you've played Jesse's game, think about implementing the *Connect4Board* class. An incomplete version is given below.

```
// fields (***** PART (a) *****)
public static final int EMPTY = ...
public static final int RED = ...
public static final int BLUE = ...
/** more fields here as needed */

// constructor
public ConnectFourBoard(int numRows, int numCols) {
    // ***** PART (b) *****
}

public boolean isFull() {
    // ***** PART (c) *****
}

public int dropPieceInColumn(int color, int col) {
    // ***** PART (d) *****
}
```

### Part (a): Fields

A *Connect4Board* object will represent one gameboard. So the class needs to have a field for the board with information about what color (red or blue) game piece is at each spot on the board, or if that spot is empty. It also needs to define the constants *EMPTY*, *RED* and *BLUE*, to be used by the other Connect-4 classes.

Think about how to store this information, then complete the “fields” part of the *Connect4Board* class.

### **Part (b): Constructor**

Write the *Connect4Board* constructor.

### **Part (c): Tie Game!**

In Connect-4, the game is considered a tie if the entire board is full of chips and no one has four pieces in a row. The *isFull* method should return false if there are any empty spots left and true if every spot on the board is filled. Write that method next.

### **Part (d): Make One Move**

As you saw when you played Jesse's version of the Connect-4 game, each player selects a column by clicking anywhere in that column. If the column isn't full, their chip "drops" into that column. The *dropPieceInColumn* method helps implement one player's move. It has two parameters: *color*, the current player's color, and *col*, the column where the user clicked. It should return -1 if the entire column is full. Otherwise it should place a chip in the correct row of the column, and return that row number.

Note that the *color* parameter should be either RED or BLUE (i.e., one of the constants defined in the *Connect4Board* class).

Write the *dropPieceInColumn* method.

### **Part (e): Compile and Run**

Now compile your code and run it using the *.class* files for the other Connect-4 classes.

## Exercise 2: Logical Thinking

**Puzzle 1:** Alice and Bob are on separate islands. Bob is sick, and Alice has the medicine. Eve has a boat and a chest that can be locked. She is willing to transport objects between Alice and Bob, but only in the chest, and if the chest is unlocked, she will steal whatever is inside. If both Alice and Bob have a padlock and a key such that their own key only opens their own lock, how can Alice send Bob the medicine so that Eve won't steal it?

**Puzzle 2:** You are on a quiz show, and you're told to make a statement. If the statement is true, you get exactly \$10. If the statement is false, you get either less than or more than \$10 but not exactly \$10. What statement can you make to guarantee that you'll get \$1,000,000?

**Puzzle 3:** Three identical airplanes start at the same airport. Each plane has a full fuel tank holding just enough fuel to allow the plane to travel half the distance around the world. These airplanes possess the special ability to transfer fuel between their tanks in mid-flight. Devise a scheme that will allow one airplane to travel all the way around the world without ever landing to get more fuel.

## Exercise 3: Practice with Exceptions

Divide into teams to play a game like Jeopardy involving questions about the code on the next page. For each question, the team that grabs the no-longer-flashing-light thingy first gets to try to answer the question. If the answer is right, the team gets points, and gets to choose the point value of the next question (100, 200, or 300 points).

```

public class ExceptionExample {

    public static void main(String[] args) {
        foo();
        try {
            bar();
        } catch (IndexOutOfBoundsException iobe) {
            System.out.println("ERROR 1");
        } catch (ArithmeticException ae) {
            System.out.println("ERROR 2");
        } finally {
            System.out.println("DONE");
        }
    }

    public static void foo() {
        // PROGRAM LINE A
        try {
            // PROGRAM LINE B
        } catch (NullPointerException npe) {
            System.out.println(npe.getMessage());
            npe.printStackTrace();
        }
    }

    public static void bar() {
        // PROGRAM LINE C
        try {
            // PROGRAM LINE D
            methodX();
        } catch (NullPointerException npe) {
            System.out.println("ERROR 3");
        } catch (IndexOutOfBoundsException iobe) {
            System.out.println("ERROR 4");
        }
        System.out.println("DONE BAR");
    }

    public static void methodX() {
        // PROGRAM LINE E
        try {
            // PROGRAM LINE F
        } catch (IndexOutOfBoundsException iobe) {
            System.out.println(iobe.getMessage());
            iobe.printStackTrace();
        } catch (NumberFormatException nfe) {
            System.out.println("ERROR 5");
            return;
        } finally {
            System.out.println("DONE METHODX");
        }
    }
}

```

## 100 point questions for Exceptions

Q: What happens if *throw new NumberFormatException()* replaces line A?

A: A stack trace would be printed, and the program would crash.

Q: What happens if *throw new NumberFormatException()* replaces line B?

A: A stack trace would be printed, and the program would crash.

Q: What happens if *throw new NumberFormatException()* replaces line C?

A: DONE would be printed, then a stack trace, and the program would crash.

Q: What happens if *throw new NumberFormatException()* replaces line D?

A: DONE would be printed, then a stack trace, and the program would crash.

Q: What happens if *throw new NullPointerException()* replaces line C?

A: DONE would be printed, then a stack trace, and the program would crash.

Q: What happens if *throw new NullPointerException()* replaces line D?

A: The following would be printed:

```
ERROR 3  
DONE BAR  
DONE
```

Q: What happens if *throw new IndexOutOfBoundsException()* replaces line A?

A: A stack trace would be printed, and the program would crash.

Q: What happens if *throw new IndexOutOfBoundsException()* replaces line B?

A: A stack trace would be printed, and the program would crash.

Q: What happens if *throw new ArithmeticException()* replaces line A?

A: A stack trace would be printed, and the program would crash.

Q: What happens if *throw new NullPointerException()* replaces line A?

A: A stack trace would be printed, and the program would crash.

## 200 point questions for Exceptions

- Q: What happens if *throw new ArithmeticException()* replaces line B?
- A: A stack trace would be printed, and the program would crash.
- Q: What happens if *throw new ArithmeticException()* replaces line C?
- A: The following would be printed:   ERROR 2  
                                          DONE
- Q: What happens if *throw new ArithmeticException()* replaces line D?
- A: The following would be printed:   ERROR 2  
                                          DONE
- Q: What happens if *throw new IndexOutOfBoundsException()* replaces line C?
- A: The following would be printed:   ERROR 1  
                                          DONE
- Q: What happens if *throw new IndexOutOfBoundsException()* replaces line D?
- A: The following would be printed:   ERROR 4  
                                          DONE BAR  
                                          DONE
- Q: What happens if *throw new NumberFormatException()* replaces line E?
- A: DONE would be printed, then a stack trace, then the program would crash
- Q: What happens if *throw new NullPointerException()* replaces line B?
- A: The NullPointerException message would be printed, then a stack trace, then the following:  
                  DONE METHODX  
                  DONE BAR  
                  DONE
- Q: What happens if *throw new NullPointerException()* replaces line E?
- A: The following would be printed:   ERROR 3  
                                          DONE BAR  
                                          DONE
- Q: What happens if *throw new NullPointerException()* replaces line F?
- A: The following would be printed:  
                  DONE METHODX  
                  ERROR 3  
                  DONE BAR  
                  DONE

### 300 point questions for Exceptions

Q: What happens if *throw new IndexOutOfBoundsException()* replaces line E?

A: The following would be printed:

```
ERROR 4  
DONE BAR  
DONE
```

Q: What happens if *throw new IndexOutOfBoundsException()* replaces line F?

A: The `IndexOutOfBoundsException` message would be printed, then the stack trace, then:

```
DONE METHODX  
DONE BAR  
DONE
```

Q: What happens if *throw new ArithmeticException()* replaces line E?

A: The following would be printed:

```
ERROR 2  
DONE
```

Q: What happens if *throw new ArithmeticException()* replaces line F?

A: The following would be printed:

```
DONE METHODX  
ERROR 2  
DONE
```

Q: What happens if *throw new NumberFormatException()* replaces line F?

A: The following would be printed:

```
ERROR 5  
DONE METHODX  
DONE BAR  
DONE
```

Q: What happens if no lines are replaced with code that throws exceptions?

A: The following would be printed:

```
DONE METHODX  
DONE BAR  
DONE
```

Q: Replacing what lines with *throw new IndexOutOfBoundsException()* would cause message "ERROR 1" to be printed (ignore other things that might be printed as well)?

A: Line C only