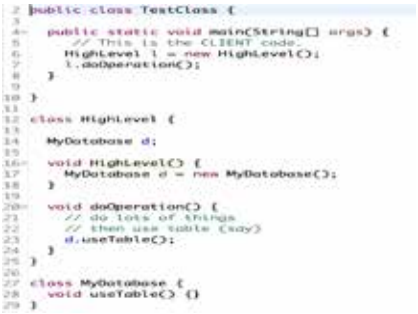
COMS 309 Week#12 Fri Worksheet-1

TEAM#? _____

NET-IDs of students who are present: _____

1 Design



```
public class TestClass {

  public static void main(String[] args) {
    // This is the CLIENT code.
    HighLevel l = new HighLevel();
    l.doOperation();
  }
}

class HighLevel {

  MyDatabase d;

  void HighLevel() {
    MyDatabase d = new MyDatabase();
  }

  void doOperation() {
    // do lots of things
    // then use table (say)
    d.useTable();
  }
}

class MyDatabase {
  void useTable() {}
}
```

Rewrite code to INJECT database dependency into the HighLevel module (from the main method). Show ALL the changes needed. *SHOW YOUR NEW CODE.*

Goal is to remove "hard link" between HighLevel class and MyDatabase.

Step 1: Create an interface for databases and have MyDatabase implement this interface.

```
interface DB {
  void useTable();
}
```

```
class MyDatabase implements DB {
        void useTable() { // code }
}
```

Step 2: a) Make HighLevel class use this interface. b) there should be NO references to MyDatabase.

```
class HighLevel {
   DB d;

   HighLevel (DB d) {
      this.d = d;
   }

   void doOperations() {
      d.useTable();
    }
```

Step 3: Application (TestClass here) should create and INJECT the MyDatabase into the HighLevel class.

```
DB d = new MyDatabase();
HighLevel h = new HighLevel(d);    //inject
h.doOperations();
```

```
public class Game {

// check if args[2] is an int
public static boolean CheckInt(String input) {
 boolean valid = true;
 for (int i = 0; i < input.length(); i++) {
  String secondarg = input.substring(i, i + 1);
  if (secondarg.compareTo("0") < 0 || secondarg.compareTo("9") > 0) {
   valid = false;
   System.out.println("First argument must be integer");
   System.exit(1);
  }

 }
 return valid;
}

public static void main(String[] args) {
 BufferedReader reader = null;
 if (args.length != 2) {
  System.out.println("Please enter two arguments");
  System.exit(2);
 }

 boolean result = CheckInt(args[1]);

 // try to open file
 try {
  reader = new BufferedReader(new FileReader(args[0]));
 } catch (FileNotFoundException fnfe) {
  System.out.println("Error opening file" + args[0]);
  System.exit(3);
 }
 boolean done = false;
 String inputLine = null;
 String[] words = null;
 int length = 0;

 while (!done) {
  try {
   inputLine = reader.readLine();
  } catch (IOException ioe) {
   System.out.println("I/O error");
   System.exit(4);
  }

  //end of file
  if (inputLine == null) {
   done = true;
  } else {
   String line = inputLine;
   String delimter = " ";
   words = line.split(delimter);
   length = words.length;
  }
 }

 //random number generator to select word from array
 Random generator = new Random();
 int num = generator.nextInt(length - 0) + 0;

 //store chosen word
 String chosen = words[num];

 //convert string to char array
 char[] chosenarr = chosen.toCharArray();
 char[] output = new char[chosenarr.length];

 for (int x = 0; x < output.length; x++) {
  output[x] = '*';
 }
 System.out.println(new String(output));

 //user guesses
 int count = 0;
 int guessnum = Integer.parseInt(args[1]);

 BufferedReader userguess = new BufferedReader(new InputStreamReader(System.in));
 String guess = null;
 char[] userlinechar = null;
 String userlinestring = null;
 String b = null;
 String c = null;

 for (int a = 0; a < guessnum; a++) {

  System.out.println("Guess a character..");

  try {

   //convert bufferedreader to chararray
   guess = userguess.readLine();
   userlinechar = guess.toCharArray();
   userlinestring = Character.toString(userlinechar[0]);

   //check if user guess is valid
   if (!userlinestring.matches("[A-Za-z]+") || userlinechar.length != 1) {
    System.out.println("Enter valid input");
    a--;
   }

   //compare user guess with chosen word
   for (int x = 0; x < chosenarr.length; x++) {


    for (int j = 0; j < output.length; j++) {
     c = new String(chosenarr);
     b = new String (output);

     if (Character.toLowerCase(userlinechar[0]) == Character.toLowerCase(chosenarr[x])) {
      output[x] = Character.toLowerCase(userlinechar[0]);

     }

    }

    }

    if (b.equalsIgnoreCase(c)){
   break;
   }

  } catch (IOException ioe) {
   System.exit(4);
  }
 }

 System.out.println(new String(output));

 // check if user has won
 int countLose = 0;

 for (int i = 0; i < output.length; i++) {
  if (output[i] == '*') {
   countLose++;
  }
 }

 if (countLose > 0) {
  System.out.println("Hard luck");
 } else {
  System.out.println("Well done");
 }
}
}
```

Redo above using MVC OR MVP. Show CODE for your classes.

IGNORE THIS ONE FOR THIS MOCK EXAM. WE DO A SIMILAR ONE IN MOCK2 THAT IS DOABLE.

2 Testing

```
public int binarySearch(int[] inputArr, int key) {

    int start = 0;
    int end = inputArr.length - 1;
    while (start <= end) {
        int mid = (start + end) / 2;
        if (key == inputArr[mid]) {
            return mid;
        }
        if (key < inputArr[mid]) {
            end = mid - 1;
        } else {
            start = mid + 1;
        }
    }
    return -1;
}
```

Q: Consider the above code. Generate test cases for it using the following techniques (first write a brief description of the technique):

a) **Boundary-value testing**

Makes sure that the boundaries of input are tested.

1. **[], 1000 // empty array**
2. **[….], 1000 // very large array (say like a MB of data)**
3. **[MININT,….], 1000**
4. **[…., MAXINT], 1000**
5. **[….], MININT**
6. **[…], MAXINT**

If you give five test cases, that is good.

b) **Equivalence-class testing**

Not the best problem to try EC testing. One way is invalid and valid data.

Note that a) EC classes must not overlap. B) Union of EC classes must equal the universe of possible testcases. Goal of EC Testing is to reduce number of test cases and yet not affect quality of testing.

Lets do this in Mock2.

c) **Statement coverage**
Makes sure that each instruction has been executed at least once.
I will normally give a smaller example.
Three test cases will do the trick here.
1. **[1,2,3], 2**
2. **[1,2,3], 1**
3. **[1,2,3], 100**

d) **Decision coverage**
Makes sure that each of the true and false paths of each decision has been taken.
It so happens that the above three test cases also cover each decision! Note that that is not always the case (We will show one in Mock2).

Q: For each step in testing, write down how it can be replaced/automated:

a) **Generating test cases.**
Using random test generation technique.

b) **Figuring out expected results.**
Instead use oracle that takes in actual input and actual results and then CHECKS if the result is ok.

c) **Writing code for test cases.**
Use parameterized testing.

d) **Running test cases.**
Drivers (junit test runners) already do this for us.

Q: What is regression testing? Explain why it is important to do regression testing.

When a change is made to software (to fix bugs, to add new features etc), the change often breaks things that used to work. Regression testing are tests that are executed to ensure that you have not broken previously working code. The goal is to make sure that your code has not "regressed" or "gone backwards". Essentially it means re-running existing test cases on your code (in addition to the new tests for the changes).

It is important to do regression tests so that one does not ship broken code to customers. Customers lose trust on updates that break existing features. It is also really expensive to respond quickly to all the customers and quickly fix and ship updates. Once a broken update is

shipped to customers, it basically creates a very bad situation for the company.

Q: What is a driver? What is a stub? What are mock objects?

Drivers "calls" SUT (System under test). They take the place of higher level modules that call the SUT.

SUT "makes calls to" stubs. Stubs take place of lower level modules that are called by the SUT. They usually respond like the "real modules" for a very limited set of inputs.

Mock objects use mocking libraries to provide stub-like behavior for testing SUT. Mock objects are also able to record the pattern of calls made to them and therefore can be used for behavioral testing of the SUT.

Q: How can you drive Web Server testing?

Using tools like Selenium Webdriver testing framework (or even CURL scripts).

Q: How can you mock Web Servers in order to test Web Clients?

Using json-server tool.

Q: How can you drive Android UI testing?

Using Espresso testing framework

3 Ethics

Q. What is a way to check if an issue/decision requires an ethical treatment?

Smell Test or Des Moines Register Test or New York Times Test. Basically, is it ok for the issue and the decision made to be posted for everyone to see? Would that change the way the decision or the way the decision would be made?

Q. What are the five approaches to an Ethical Decision? Use an example to explain.

SEE SLIDES

Q. What are the eight principles of IEEE's code of conduct?

SEE SLIDES