IT 313 – Midterm Exam

February 13, 2020 **Name** _____

Part A. Multiple Choice Questions. Answer all questions. Optional: supply a reason or show work for partial credit in case you select the wrong answer. If you select the correct answer, your reason or work will not be considered. 5 points for each question.

1. Which is the header line for the Java startup method in a class?
   a. `private static void startup(String args) {`
   b. `public static void main(String args) {`
   c. `public static void main(String[ ] args) {`
   d. `public void static void StartUp(String[ ] args) {`

2. Which of these primitive datatypes takes up 4 bytes in memory?
   a. `boolean` b. `float` c. `int` d. `long`

3. Which statement converts the `double` value `x` into the `String` object `t`?
   a. `t = x.toString( );` b. `t = String.valueOf(x);`
   c. `t = Double.toString(x);` d. `t = (String) x;`

4. Which statement invokes the `toString` method in the base class B for an object?
   a. `base.toString( )` b. `B::toString( )`
   c. `super.toString( )` d. `toString.base( )`

5. A method that is defined the a derived class with the same name and signature of a method in the base class is called an _____ method.
   a. overridden b. overloaded c. protected d. static

6. What is the output?

```
for(int n = 3; n <= 20; n *= 2) {
    System.out.print(n + " ");
}
```

   a. `2 4 8 16` b. `3 6 9 12` c. `3 6 12` c. `3 6 12 24`

7. The `findMinIndex` method finds the index of the smallest item in an `int` array. In case of a tie for the smallest item, the first index of the tied items is returned.

```
public class MyClass {
    public static int findMinIndex(int[ ] array) {
        int minIndex = 0, val = array[0];
        for(int i = 1; i < array.length; i++) {
            if (array[i] < val) {
                val = array[i];
                minIndex = i;
            }
        }
        return minIndex;
    }
}
```

If the array a is defined as

```
int[ ] a = {3, 6, 5, 8, 1, 7};
```

which of these assert statements is correct for unit testing the `findMinIndex` method?

a. `assertEquals(MyClass.findMinIndex(a) == 1);`
b. `assertEquals(MyClass.findMinIndex(a), 1);`
c. `assertEquals(MyClass.findMinIndex(a), 4);`
d. `assertEquals(MyClass.findMinIndex(a[ ]), 4);`

8. The collection **`col`** is defined as

```
ArrayList<String> col = new ArrayList<String>( );
```

Which statement sets the `ArrayList` item with index 3 to `"Chicago"`?

a. `ArrayList.set(col, 3, "Chicago")`
b. `col.set(3, "Chicago");`
c. `col.set(3) = "Chicago";`
d. `col.set[3] = "Chicago";`

9. Which choice is the correct ordering of a code fragment that reads the first line of the home page of the CDM website?

a.
```
URL urlObject = new URL(urlString);
String urlString = "http://facweb.cdm.depaul.edu/sjost/it212/";
Scanner s = new Scanner(urlObject.openStream( ));
String line = s.nextLine( );
System.out.println(line);
s.close( );
```

b.
```
String urlString = "http:// facweb.cdm.depaul.edu/sjost/it212/";
Scanner s = new Scanner(urlObject.openStream( ));
URL urlObject = new URL(urlString);
String line = s.nextLine( );
System.out.println(line);
s.close( );
```

c.
```
String urlString = "http:// facweb.cdm.depaul.edu/sjost/it212/";
URL urlObject = new URL(urlString);
Scanner s = new Scanner(urlObject.openStream( ));
String line = s.nextLine( );
s.close( );
System.out.println(line);
```

d.
```
String urlString = "http:// facweb.cdm.depaul.edu/sjost/it212/";
URL urlObject = new URL(urlString);
String line = s.nextLine( );
Scanner s = new Scanner(urlObject.openStream( ));
a.close( );
System.out.println(line);
```

10. Predict the output of `Main.java`. The choices for output are shown on Page 5.

```
// ==== Source code file A.java
public class A {
    private int x;

    public A(int theX) {
        this.x = theX;
    }

    public void augment( ) {
        this.x += 3;
    }

    @Override
    public String toString( ) {
        return "%" + x + "%";
    }
}

// ==== Source code file B.java
public class B extends A {
     private int y;

     public B(int theX) {
          super(theX);
          this.y = 7;
     }

     @Override
     public void augment( ) {
          super.augment( );
          this.y *= 2;
     }

     @Override
     public String toString( ) {
          return "$" + this.y + "$" + super.toString( );
     }
}
```

```
// Use this variable trace table to
// help you predict the output on
// Page 5.

   a.x         b.x     b.y
+-------+   +-------+-------+
|       |   |       |       |
+-------+   +-------+-------+
|       |   |       |       |
+-------+   +-------+-------+
|       |   |       |       |
+-------+   +-------+-------+
```

```
// ==== Source code file Main.java
public class Main {

     public static void main(String[ ] args) {
          A a = new A(10);
          a.augment( );
          System.out.print(a.toString( ) + " ");

          B b = new B(13);
          b.augment( );
          System.out.printf(b.toString( ));
     }
}
```

a. `$10$ %13%$10$`
b. `%13% $14$%16%`
c. `$13$ %14%$16$`
d. `%13% $26$%14%`

Part B: Correct the Errors. The classes `Animal` (this page) and `Pet` (Page 6) form an inheritance hierarchy. The `Main` class (Page 7) class populates an `ArrayList` collection with Pet objects from an input file. There are about 15 total errors in these three classes. Correct these errors. Inserting, correcting, or removing a pair of `' '`, `" "`, `( )`, `[ ]`, or `{ }` only counts as one error. Also, swapping two items only counts as one error, as does making more than one correction to a variable name. Number the errors that you find to them easier to see. 20 points.

```
// ---------------------------------------------------------------
// Source code file: Animal.java
public class Animal {
    private String animalType;
    private char gender;

    public Animal(string animalType, char gender) {
        this.animalType = animal_type;
        gender = this.gender;
    }

    Override
    public String toString( ) {
        return String.format(%s %c, this.animalType, this.gender);
    }


// ---------------------------------------------------------------
```

```
// ---------------------------------------------------------------
// Source code file: Pet.java

private class Pet extends Animal:
    private String owner;
    private bool vaccinated;


    public void Pet(String animalType, char gender, String owner) {
        this.owner = owner;
        this.vaccinated = false;
        super(animalType, gender);
    }


    public int isVaccinated( ) {
        return this.vaccinated ? "Yes" : "No";
    }


    public void setVaccinated {
        this.vaccinated = true;
    }


    @Override
    public String toString( ) {
        return String.format("%s %s %s",
            super.toString( ), this.owner( ), this.isVaccinated( ));
    }
}
```

```
// ------------------------------------------------
// Source code file: Main.java

import java.io.File;
import java.io.FileNotFoundException;
import java.util;

public class Main {
    public static final int FILE_NOT_FOUND == 1;
    public static void Main(String[ ] args) {
        File f = new File("pets.txt");
        Scanner s = null;
        ArrayList<Pet> col = ArrayList<Pet>(50);
        try {
            Scanner s = new Scanner(f);
        }
        catch (FileNotFoundException) {
            System.out.println("Input file not found.");
            System.exit(FILENOTFOUND);
        }

        while(s.hasNextLine( )) {
            String line = s.nextLine( );
            String[ ] fields = line.split(",");
            String animalType = fields[0];
            char gender = fields[1].charAt[0];
            String owner = fields[2];
            col.add(new Pet(animalType, gender, owner));
        }

        System.out.println(col);
        for(Pet p : col) {
            p.setVaccinated( );
        }

        forPet p in col {
            System.out.print(p.isVaccinated( ) + " ");
        }
    }
    s.close( );
}
```

Part C: Write Getters and Setters. Write the source code for the getters and setters of the `gender` instance variable in the `Animal` class on Page 5. 10 points.

1. Write the getter for the `gender` instance variable:

2. Write the setter for the `gender` instance variable:

Part D: Convert Traditional Test to Unit Test. Convert the traditional test class `Test1` on this page to the unit test class `Test2` on Page 10. 10 points.

```
// ==== Source code file: OlympicMedal.java ==========================
public class OlympicMedal {
    private String event;
    private int medalType;

    public OlympicMedal(String event, int medalType) {
        this.event = event;
        if (1 <= medalType && medalType <= 3) {
            this.medalType = medalType;

        }
        else {
            this.medalType = 0;
        }
    }

    @Override
    public String toString( ) {
        String[ ] medalTypes = {"Unknown", "Gold", "Silver", "Bronze"}
        return String.format("%s %s",
            this.event, medalTypes[this.medalType]);
    }
}

// ==== Source code file: Test1.java ==================================

public class Test1 {
    public static void main(String[] args) {
        OlympicMedal m1 = new OlympicMedal("Archery -- Men", 2);
        System.out.println(m1);
        OlympicMedal m2 = new OlympicMedal("Marathon -- Women", 3);
        System.out.println(m2);
        OlympicMedal m3 = new OlympicMedal("Equestrian", 5);
        System.out.println(m3);
    }
}

// ====================================================================
```

```
// ==== Source code file: Test2.java ==================================

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class OlympicMedalTest {

    private OlympicMedal m1, m2, m3;

    @BeforeEach
    void setUp( ) {

        m1 = new OlympicMedal("Archery -- Men", 2);
        m2 = new OlympicMedal("Marathon -- Women", 3);
        m3 = new OlympicMedal("Equestrian", 5);
    }

    @Test
    void toStringTest( ) {

        // Write assertEquals statements here to test the
        // toString method of the OlympicMedal class for the three
        // objects m1, m2, and m2.
        // Usage:
        // assertEquals(computed, expected);









    }
}

// ====================================================================
```