ELEC 2543 Object-Oriented Programming and Data Structures
Second Semester 2016 – 2017
Mid-Term Examination 2
April 3, 2017

SOLUTION

This exam has 3 questions and 10 pages (including this one). Inform us if you find any page(s) missing.

Put your answers in the answer sheets provided.

Program codes should be indented properly to enhance readability.

All library methods can be used. If you do not remember the exact name, give the method a name and briefly explain what it does. To receive full credit, you have to show how you invoke the method.

You have to use the appropriate modifiers, return types, and parameters to define methods requested unless specified otherwise.

The maximum score of this exam is 100. **Spend your time wisely**. Good luck!

This is the cover page and there is NO question on this page.

1. Short Questions

a) (6 points) For each of the following situations, suggest which modifier should be used in the declaration.

i. Define a method that cannot be overridden by a child class
ii. Define a variable that all the objects of the same class have access to the same copy
iii. Define an instance variable in the parent class that can be accessed by the child classes but not outside classes directly

i. final
ii. static
iii. protected

b) (5 points) Use enumerated type to define type `DaysOfAWeek` for the days in a week. A week starts on Sunday.

```
enum DaysOfAWeek {
        SUN, MON, TUE, WED, THU, FRI, SAT;
}
```

c) (5 points) Write method `computeSum` that accepts a variable number of integer parameters and returns the sum of the parameters. That is,

`computeSum()` returns 0
`computeSum(1, 3, 2)` returns 6

You do not have to specify any modifier for this method. You have to use appropriate return type and parameter.

```
int computeSum(int … arr) {
  int sum = 0;
     for (int i = 0; i < arr.length; i++)
      sum += arr[i];

     return sum;
}
```

d) (6 points) For each of the declaration statements in `TestABC.main`, explain whether it is valid or not. If it is not valid, specify whether it is compilation error or runtime error. If it is valid, give the output. [You can assume each statement is executed independently.]

```
public abstract class A {
  public A() {
    System.out.println("hello from A");
  }
}

public class B extends A {
  public B() {
    System.out.println("hello from B");
  }
}

public class C extends B {
  public C() {
    super();
    System.out.println("hello from C");
  }

  public here() {
    System.out.println("here C");
  }
}

public class TestABC {
    public static void main(String args[]) {
         A a = new B();       // Statement (i)
         A b = new A();       // Statement (ii)
    }
}
```

`A a = new B();`

Statement is valid and output:

```
hello from A
hello from B
```

`A b = new A();`

Statement is not valid, Abstract classes cannot be instantiated. Compilation error

e) (8 points) Refer to the class definitions in part (c), give the output of statements

```
B c = new C();
c.here();
```

If you think the statement(s) has error, explain clearly whether it is compilation error or runtime error. [If there is error in the first statement, you do not have to provide output. If only the 2nd statement has error, provide the output of the first statement.]

The 2nd statement has error, The method `here()` is undefined for the Class `B`, compilation error.
Output of the first statement:

```
hello from A
hello from B
hello from C
```

2. (30 points) In an application, we would like to create geometry objects `Circle` and `Rectangle`. Both `Circle` and `Rectangle` are derived from class `Shape`.

   To model the behavior that an object can be resized, interface `Resizable` is defined as follows:

```
public interface Resizable {
  public void resize(double scale);
     // method to resize the object according to the scale.
     // for circles, the radius becomes radius*scale.
     // for rectangles, both the height and width are
     // multiplied by scale.
}
```

   Class `Shape`:

   - `Shape` is an abstract class and implements the `Comparable` and `Resizable` interfaces.
   - It also has an abstract method `public abstract double area()`. This method returns the area of the shape.
   - Two shapes can be compared using their areas.

   The output of a driver program is given on the next page.

   Give the definitions of `Shape`, `Circle`, and `Rectangle` that could generate the output. You should adopt an object-oriented design and use appropriate data types for all instance variables.

   To save time, you DO NOT have to define constructors for `Circle` and `Rectangle`. You can assume the constructors have been given to initialize the instance variables.

   You can assume π is `Math.PI`.

```
public class TestShape {

  public static void compare(Circle circ, Rectangle rect) {

    System.out.println("The area of the circle is " +
                                          circ.area());
    System.out.println("The area of the rectangle is " +
                                          rect.area());

    if (circ.compareTo(rect) < 0) {
      System.out.println("The circle is smaller
                                than the rectangle.");
    } else if (circ.compareTo(rect) > 0) {
      System.out.println("The circle is larger
                                than the rectangle.");
    }
  }

  public static void main(String[] args) {
    Circle circ = new Circle(3);
                // create a circle with radius 3
    Rectangle rect = new Rectangle(4, 5);
                 // create a rectangle with width 4 and height 5

    compare(circ, rect);

    circ.resize(0.8);
    rect.resize(1.1);

    System.out.println("\nAfter Resizing....");
    compare(circ, rect);
  }
}
```

The output of `TestShape.main` is:

```
The area of the circle is 28.274333882308138
The area of the rectangle is 20.0
The circle is larger than the rectangle.

After Resizing....
The area of the circle is 18.095573684677213
The area of the rectangle is 24.200000000000003
The circle is smaller than the rectangle.
```

Shape.java

```
public abstract class Shape implements Comparable, Resizable{

  public abstract double area();

  public int compareTo(Object other) {
    Shape s = (Shape) other;
     // == may not work for comparing double's
    if (Math.abs(area() - s.area()) < 0.000001) return 0;

    if (area() > s.area()) return 1;

    return -1;
  }
}
```

Circle.java

```
public class Circle extends Shape {
  private double radius;

  public Circle(double r) {
    radius = r;
  }

  public double area() {
    return radius*radius*Math.PI;
  }

  public void resize(double scale) {
    radius *= scale;
  }
}
```

Rectangle.java

```
public class Rectangle extends Shape {
  private double width;
  private double height;

  public Rectangle(double w, double h) {
    width = w;
    height = h;
  }

  public void resize(double scale) {
    width *= scale;
    height *= scale;
  }

  public double area() {
    return width*height;
  }
}
```

3. Please refer to the program listings on pages 9-10 for this question.

a) (5 points) Give the output of the `TestMyInt.main`.

[0, 4, 2, 0, 3]
[0, 1, 4, 3, 4]

b) (5 points) Define the `equals` method for class `MyInt` that returns `true` if the objects both carry the same numeric value in instance variable `num`.

```
public boolean equals(MyInt other)
{
      return this.num == other.num;
}
```

In the following parts, you can assume the `equals` method has been properly defined in class `MyInt`.

c) (5 points) Identify `i` and `j` such that:

```
MyInt num1 = intArr[i];
MyInt num2 = intList.get(j);
```

`num1 == num2` returns `false` but `num1.equals(num2)` is `true`.

If you think there is no such `i` and `j` exist, please say so. If you think there are more than one `i` and `j`, please say so as well.

there is such i and j exist, i=1 and j =4.
when i=1 and j =4, num1 and num2 are not the same object, so num1 == num2 returns false, but num1.equals(num2) just compares member variable num, num1.num = 4 and num2.num =4 so returns true

d) (5 points) Give the output of the following codes. You should assume they are put after the codes in part (a).

```
for (MyInt i : intArr)
    i.increment();

printArray(intArr);
printList(intList);
```

[2, 5, 3, 2, 4]
[2, 1, 5, 4, 4]

e) (5 points) Give the output of the following codes. You should assume they are put after the codes in part (a) without the codes in part (d).

```
intArr[0] = MyInt.copy(intArr[0]);
increment(intArr[0].intValue());
intArr[3].increment();

printArray(intArr);
printList(intList);
```
[0, 4, 2, 1, 3]
[1, 1, 4, 3, 4]

f) (10 points) Develop method `removeListDuplicate` in class `TestMyInt` that removes elements in `intList` that carries the same as numeric value as at least one element in `intArr`. That is, if

`intArr = [9, 9, 3, 8, 1]` and `intList = [2, 3, 10, 9, 8]`

After calling the method, `intList` becomes `[2, 10]`. `intArr` remains unchanged in the process.

You cannot create any new `MyInt` object but you can create new `MyInt[]` or `ArrayList<MyInt>` if needed.

```
private static void removeListDuplicate() {

    ArrayList<MyInt> newList = new ArrayList<MyInt>();

    for (int i = 0; i < intList.size(); i++) {

      MyInt num = intList.get(i);

      boolean duplicate = false;

      for (int j = 0; j < intArr.length; j++) {
        if (num.equals(intArr[j]))
          duplicate = true;
      }

      if (duplicate == false)
        newList.add(num);
    }

    intList = newList;
  }
```

g) (5 points) Use your implementation of `removeListDuplicate`, give the output of the following codes. You should assume they are put after the codes in part (a) without the codes in parts (d) & (e).

```
<calling removeListDuplicate>
printArray(intArr);
printList(intList);
```

Explain whether there is any `MyInt` garbage created in `removeListDuplicate`.

[0, 4, 2, 0, 3,]
[1]
A garbage is created. `MyInt` object that was added as the last element in `intList` with `num`=4 has become a garbage.

The following two pages contain the program listings for Question 3. Detach this sheet of paper as needed to ease reading.

```
public class MyInt {
  private int num;

  public MyInt(int num) {
    this.num = num;
  }

  public int increment() {
    num++;
    return num;
  }

  public int intValue() {
    return num;
  }

  public MyInt copy() {
    return this;
  }

  public static MyInt copy(MyInt other) {
    return new MyInt(other.num);
  }

  public String toString() {
    return Integer.toString(num);
  }
}
```

```
public class TestMyInt {

  private static MyInt[] intArr;
  private static ArrayList<MyInt> intList;


  private static void printArray(MyInt[] arr) {
        // method that prints array elements in the format of
        // [arr[0], arr[1], arr[2], …, ]
  }

  private static void printList(ArrayList<MyInt> intList) {
      System.out.println(intList);
  }

  private static void increment(int i) {
    i++;
  }

  public static void main(String[] args) {
    intArr = new MyInt[5];

    intArr[0] = new MyInt(0);
    intArr[1] = new MyInt(4);
    intArr[2] = new MyInt(2);
    intArr[3] = intArr[0].copy();
    intArr[4] = new MyInt(3);

    intList = new ArrayList<MyInt> ();

    intList.add(intArr[3]);
    intList.add(new MyInt(1));
    intList.add(intArr[1]);
    intList.add(intArr[4]);
    intList.add(new MyInt(4));

    printArray(intArr);
    printList(intList);

    // end of the codes in part (a)

  }

}
```