ELEC 2543 Object-Oriented Programming and Data Structures
Second Semester 2016 – 2017
Mid-Term Examination
February 20, 2017
SOLUTION

This exam has 3 questions and 9 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.

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) (5 points) Write a single statement that defines an integer constant `RATE` that is equal to 2.

```
final int RATE = 2;
```

b) (5 points) Alice is developing a Java class to represent rolling dies. Random numbers are needed in various methods in her class. She develops the following method in the class:

```
int genRanNum() {
     return (int) (Math.random()*6) + 1;
}
```

What visibility modifier should she use for this method? Briefly explain why.

Private. genRanNum method will be used within the class only. It is not necessary and appropriate for a method to be declared as public if only methods inside the same class will use it.

c) (5 points) Suppose you want to develop a class that represents the shape rectangle. Give two attributes and one method that you would define for its behavior. You only have to give the method header and explain what it is. You do not have to provide the definition. The method should not be constructor, `toString()`, or getters/setters.

```
public class Rectangle {
   private int height;   // height of the rectangle
   private int width;    // width of the rectangle
   public int getRectangleArea(){   // returns the area
   }
}
```

d) (10 points) Give the output of the above codes.

```
String s1 = s2 = "ELEC2543";
s1 = s2.replace('X', 'x'); // replace all 'X' to 'x' in the String

System.out.println("s1 = " + s1);
System.out.println("s2 = " + s2);
```

Output:

```
s1 = ELEC2543
s2 = ELEC2543
```

How many `String` objects have been created? Is there any garbage? Explain your answers.

2 String objects have been created.
No garbage, s1 and s2 both refer to the same object in the beginning. In `s1 = s2.replace("X", "x")`, new object is created and assigned to s1, but s2 still refers to the original object, so no garbage.

e) (5 points) Given the following statement:

```
String s = "A piece of Cake!";
```

Write the fragment of codes that first prints out the string in uppercase and then the original string. You cannot declare another `String` variable.

```
System.out.println(s.toUpperCase());
System.out.println(s);
```

f) (5 points) Are `obj1` and `obj2` pointing to the same Integer object? Explain your answer.

```
Integer obj1, obj2;
obj1 = 100;
obj2 = new Integer(100);
```

They are not pointing to the same Integer object.
In `obj1 = 100`; autoboxing occurs and obj1 will be pointing to the Integer object resulted in autoboxing.
`obj2 = new Integer(100)` will create a new Integer Object, not by autoboxing.

2. Question 2 is based on the following class definition.

```
// a class definition for complex numbers

public class Complex {

    private int real;
    private int imag;


    // constructor
    public Complex (int real, int imag) {
         this.real = real;
         this.imag = imag;
    }


    // copy the instance variable values from c
    public void copy(Complex c) {
         real = c.real;
         imag = c.imag;
    }


    // make a copy of the Complex object
    public Complex copy() {
         return new Complex(real, imag);
    }


    public String toString() {
          // the Complex number is printed as a+bi
          // where a is the real part and
          // b is the imaginary part
          // e.g. if real = 3 and imag = 4
          // the complex number is printed as 3+4i
    }

}
```

a) (5 points) The *conjugate* of the complex number a + b*i* is a – b*i*. (For example, the conjugate of *3+4i* is *3-4i*.) Write a method to be put inside class `Complex`

```
public Complex conjugate()
```

that returns a `Complex` object representing the conjugate of the `Complex` object invoking the method.

```
public Complex conjugate(){
        return new Complex(real, -1*imag);
}
```

b) (5 points) Develop the following method in class `Complex`

```
public boolean theSameAs(Complex c)
```

This method returns true only if the `Complex` object that invokes the method has the same instance variable values as the parameter `c`.

```
public boolean theSameAs(Complex c){
    return real == c.real && imag == c.imag;
}
```

c) (10 points) Based on the method definitions defined in Questions (a) – (b), give the output of the following program:

```
Complex c1 = new Complex(3, 4);                    // Line 1
Complex c2 = new Complex(5, 6);          // Line 2
Complex c3 = c1.conjugate();             // Line 3

System.out.println("Section 1");
System.out.println(c1);
System.out.println(c2);
System.out.println(c3);


Complex c4 = c1;                    // Line 4
c1.copy(c2);                        // Line 5
System.out.println("Section 2");
System.out.println(c1);
System.out.println(c2);
System.out.println(c3);
System.out.println(c4);


c2 = c3.copy();                           // Line 6
c1 = c3.conjugate();                      // Line 7
System.out.println("Section 3");
System.out.println(c1);
System.out.println(c2);
System.out.println(c3);
System.out.println(c4);
```

```
Section 1
   3+4i
   5+6i
   3-4i
Section 2
   5+6i
   5+6i
   3-4i
   5+6i
Section 3
   3+4i
   3-4i
   3-4i
   5+6i
```

d) (10 points) For each line in Lines 1 – 7, explain whether there is any object created or become garbage.

Line 1: object is created and no garbage
Line 2: object is created and no garbage
Line 3: object is created and no garbage
Line 4: no object is created and no garbage
Line 5: no object is created and no garbage
Line 6: a new object is created and the original object c2 referred to becomes a garbage
Line 7: a new object is created and no garbage

3. In Game LuckyDraw, there is a bank and some players. The bank initially has a certain amount of money, while all players do not have any money. In each round, a player is selected at random to take a random amount of money from the bank. In this way, at least one player gets some money in each round. The game ends when the bank runs out of money.

   The following is a possible game of two players with initial bank cash of 10 dollars.

```
The bank has 10 dollars.
Player 1 has 0 dollars.
Player 2 has 0 dollars.
Player 1 draws 3 dollars from the bank.

The bank has 7 dollars.
Player 1 has 3 dollars.
Player 2 has 0 dollars.
Player 2 draws 5 dollars from the bank.

The bank has 2 dollars.
Player 1 has 3 dollars.
Player 2 has 5 dollars.
Player 2 draws 2 dollars from the bank.

The bank has 0 dollars.
Player 1 has 3 dollars.
Player 2 has 7 dollars.
```

   The bank balance cannot be negative. That is, if the bank has 2 dollars left, a player can at most get 2 dollars.

   Develop the game with two players according to the following.

Class `Player` represents the players in the game. The following has been defined.

```
public class Player {


  private int cash;  // current amount of money the player has
  private int id;    // id of the player


  // constructor
  // every player starts with 0 cash
  public Player(int id) {
    cash = 0;
    this.id = id;
  }


  // getter method of instance variable cash
  public int getCash() {
    return cash;
  }


  // getter method of instance variable id
  public int getID() {
    return id;
  }


}
```

a) (5 points) Develop method `public int addCash(int amt)` in class `Player` that adds `amt` dollars to the player that calls this method. The method returns the amount of money the player has after adding the money.

```
public int addCash(int amt)
  {
     cash += amt;
     return cash;
  }
```

Class `LuckyDrawGame` simulates the game of two players.

```
public class LuckyDrawGame {

    private int cashAmt;    // amount of money in the bank
    private Player p1, p2;

}
```

b) (5 points) Develop the constructor for class `LuckyDrawGame` that accepts a single integer as the parameter. The parameter is the initial amount of cash in the bank. The two players should be initialized with id 1 and 2 in the constructor.

```
public LuckyDrawGame(int cash)
  {
        cashAmt = cash;
        p1 = new Player(1);
        p2 = new Player(2);
  }
```

Suppose that the following methods are defined in `LuckyDrawGame`.

Method `private Player findPlayer()` returns either `p1` or `p2` at random.

Method `private void printStatus()` prints out the money the bank and each player has. For example, if the method is called before any player draws money, the output of the bank with 10 dollars in the beginning is

```
The bank has 10 dollars.
Player 1 has 0 dollars.
Player 2 has 0 dollars.
```

c) (15 points) Develop method `public void play()` in `LuckyDrawGame` that simulates the game and produces the output. You can use any method developed or provided in (a)-(b) but not necessary.

The skeleton of method `play()` is as shown. You can define other local variables. However, you cannot define other instance variables in class `LuckyDrawGame`.

```
public void play () {

    Die die = new Die();
    // a die the returns a random number from 1 to 6
    // when method roll() is called.
    // [1, 6] is the amount of money a player
    // can take from the bank each time (if possible)

    do {

        // put the codes needed here for completing
        // the game on your answer sheets

        // the status of the bank and each player must
        // be printed in each round

        // a message telling which player draws money
        // must be printed as shown in the sample output


    } while (cashAmt > 0);
    printStatus();

}
```

```
        do {
          printStatus();
          int cashValue = die.roll();
          Player p = findPlayer();

          if(cashValue <= cashAmt)
          {
                p.addCash(cashValue);
                cashAmt -= cashValue;
          }
          else
          {
                p.addCash(cashAmt);
                cashValue = cashAmt;
                cashAmt = 0;
          }
          System.out.println("Player " + p.getID() +
                " draws " + cashValue + " dollars from bank.");
        } while (cashAmt > 0);
        printStatus();
    }
```

d) (10 points) Suppose now you want to enhance the game to be played by three players with id 1, 2, and 3. Explain which parts of `LuckyDrawGame` have to be changed and how to change them (with code fragments for illustration).

If you use method `findPlayer()` or `printStatus()`, you only have to briefly describe what changes are needed there. For example, the following description would be sufficient for `printStatus()`.

"Method `public void printStatus()` should print out the status of Player 3 as well."

```
public class LuckyDrawGame {

        private int cashAmt;    // amount of money in the bank
        private Player p1, p2;

    }
```

(1) One more Player instance variable p3 should be defined.
private Player p1, p2, p3;
(2) Instance variable p3 should be initialized in the constructor: p3 = new Player(3);
(3) Method findPlayer() should return Player 1, Player 2, Player 3 at random.
(4) Method printStatus() should print out the status of Player 3 as well.