BBM102 – Introduction to Programming II – Spring 2016

2. Ara Sınav – 2nd Midterm Exam
27.04.2015

KEY

Name Surname:_____

Student ID :_____ **Section:**_____

Question / Soru	1	2	3	4	5	6	7	8	Toplam / Total
Points / Puan	10	10	12	14	14	10	16	14	100
Grade / Not									

Question 1.
Would the following code compile, and if it does what would be the output? Answer for the two cases given below;

```
public class Foo {
      public static void boom() throws MyException {
            System.out.println("Boom Boom");
      }

      public static void main(String[] args) {
            boom();
            System.out.println("Bang Bang!");
      }
}
```

a) MyException is extending Exception

Compiler Error, Exception is checked exception so compiler forces us to handle it.

b) MyException is extending RuntimeException

Boom Boom
Bang Bang!

Question 2.
Given that the following main method compiles and outputs "false" draw the UML Class Diagram showing the hierarchy.

```
public static void main(String[] args) {
            C[] cs = new C[4];
            cs[0] = new A();
            cs[1] = new B();
            cs[2] = new C();
            cs[3] = new D();

            A a = new D();
            System.out.println(cs[1] instanceof A);
}
```

Two possibilities:

C <- B <- A <- D
or
C <- A <-D
  <- B

Question 3.
What will be the output of the program below?

```
public class MyList {
  public static void main(String[] args) {
    List<Integer> l1 = Arrays.asList(1, 3, 5, 7);
    List<Integer> l2 = Arrays.asList(5, 7, 9, 11);

    ArrayList<Integer> l3 = new ArrayList<Integer>(l1);
    ArrayList<Integer> l4 = new ArrayList<Integer>(l2);

    ArrayList<Integer> l5;
    ArrayList<Integer> l6;

    l5 = l3;
    l6 = new ArrayList(l4);

    l3.remove(0);
    l4.remove(0);
    l5.remove(0);
    l6.remove(0);
    System.out.println(l3);
    System.out.println(l4);
    System.out.println(l5);
    System.out.println(l6);
   }
}
```

5, 7
7, 9, 11
5, 7
7, 9, 11

Question 4.

```
public abstract class ClassA {
    int var = 1;
   public ClassA() { }
   public ClassA(String a) { }

   abstract public void methodA();

   public void methodA2(){
     System.out.println("this is classA.");
   }
}
```

```
public abstract class ClassB extends ClassA {

      public void methodB(){
            System.out.println("this is classB.");
      }
}
```

```
public class ClassC extends ClassB
implements I2{

}
```

```
public interface I2 extends I1{

}
```

```
public interface I1 {
   int a;
   public I1(){};
   void method1();
}
```

This code has some (syntax) errors. Which of the following must be done in order to correct these errors? Explain the reason for each correction with only one sentence.

a) The constructors must be deleted in ClassA

b) The variable var must be deleted in ClassA.

c) The variable var must be defined static in ClassA.

d) The methodA2() must not be concrete; it must be defined abstract and its body must be deleted.

e) The methodB() must be defined abstract in ClassB and its body must be deleted.

f) The constructor in interface I1 must be deleted.

g) The method1() must be implemented in I2.

h) The variable a must be defined static in I1.

i) The variable a must be initialized in I1.

j) The methodA() must be implemented in ClassC.

k) The method1() must be implemented in ClassC.

Question 5.

```
public class ExceptionA extends
Throwable {}
```

```
public class ExceptionB extends
ExceptionA{
      public String toString(){
            return "Ted bread, ";
      }
}
```

```
public class X
{
    public static void main(String [] args)
    {
      try{
            try {
                  badMethod();
                  System.out.print("Fear ");
              }
              catch (ExceptionB ex) {
                  System.out.print("Fred ");
                  throw new ExceptionB();
              }
              catch (ExceptionA ex1) {
                  System.out.print("tear ");
              }
              finally {
                  System.out.print("fed ");
              }
      }
      catch(ExceptionA e) {
            System.out.print(e);
      }
      finally {
            System.out.print("Ted fed ");
      }
      System.out.print("Fred bread.");
    }
    public static void badMethod () throws ExceptionB {
        throw new ExceptionB();
    }
}
```

Print the output of the program above.

Fred fed Ted bread, Ted fed Fred bread.

Question 6.
Consider the code below. Does it compile/work? Try to answer by considering the possible ambiguity in the code. Ambiguity means that something is not deterministic, extra information is needed to determine the correct behavior; i.e. we do not know how it will work. If your answer is "yes it compiles", why do you think the code is referred to as ambiguous? And, why there is no ambiguity? If your answer is "no it does not compile", what is the problem with the code?

```
public interface I1 {
      void ambiguousMethod();
}
```

```
public interface I2 {
      void ambiguousMethod();
}
```

```
public class AmbiguousCall implements I1, I2 {
       public void ambiguousMethod() {
            System.out.println("I am ambiguous?");
      }
}
```

```
public class AmbiguityCheck {
      public static void main(String[] args) {
                  AmbiguousCall a = new AmbiguousCall();
                  a.ambiguousMethod();
      }
}
```

It compiles. Interfaces do not define an implementation, so although the same method signature contract is assigned to the Class, the inherited implementation is not ambiguous.

Question 7.

Write a Java program that takes N integers from the user. This program will output integers with a repeated magnitude. For example, 5 and -5 have the same magnitude.

When user enters the value 0 the program should terminate (see example below). The number of integers that will be entered is not constant and you are expected to use an appropriate Collections class in your program to store and process the integers.

Example run of the program - 1:

>1 -1 5 6 7

1

Example run of the program -2 :

>3 -1 -3

3

Example run of the program - 3:

> 5 -10 -5 8 -2 9 -5 8 -12 -8 7 0

5
5
8
8

```
Scanner newScanner = new Scanner(System.in);

HashSet set = new HashSet();
int val =  newScanner.nextInt(); //reads one integer

while(val!=0){
      int magnitude;
      if(val<0)
        magnitude = -val;
      else
        magnitude = val;

      if(set.contains(magnitude))
        System.out.println(magnitude);
      else
        set.add(magnitude);

       val =  newScanner.nextInt();
}
```

Question 8.
There are 5 compiler errors in the following classes and interfaces. Mark and shortly explain these errors.

```
public interface IMove
{
   public void
twist();
}
```

```
public interface IDance implements
IMove {
      public void shake();
      public abstract void swing();
}
```

```
public abstract class Dance implements IDance {
  public final void shake() {
      System.out.println("Shake!");
  }
}
```

```
public final class Mumba extends Dance {
     public void swing() {
          System.out.println("The Mambo Swing");
     }
}
```

```
public class Rumba implements Dance {
     public final void shake() {
          System.out.println("the Rumba Shake!");
     }
     public abstract void twist() {
          System.out.println("the Rumba twist");
     }

}
```

1 – IDance should extend IMove, an interface can only extend another Interface

2 – Rumba should extend Dance, a class can only extend another class

3 – final method shake cannot be overridden in Rumba

4 – twist() in Rumba cannot be abstract, must remove abstract

5 – Rumba should implement swing, Mumba should implement twist