BBM102 – Introduction to Programming II – Spring 2016

Final Exam
25.05.2016
Duration: 90 minutes

Name Surname: ____

Student ID : ______________________ **Section:**_______________

Question	1	2	3	4	5	6	7	8	Total
Points	**12**	**10**	**14**	**16**	**10**	**10**	**14**	**14**	**100**
Grade									

Question 1.

For the given class definitions below, determine if the following expressions will run "**without any errors**" or will result in a "**compile time**" **/** "**runtime**" **error**. For the expressions with errors, explain clearly what the error is and explain why it is an error.

```
public class A {

}
```

```
public class B extends A{

}
```

```
public class C extends A{

}
```

```
public class D extends B{

}
```

```
public class E extends D{

}
```

a) `B b = new D();`
No error

b) `A a = (D) new B();`
Runtime error (B cannot cast to D)

c) `D d = ((B) new A());`
Compile error (cannot convert from B to D)

d) `A a = (B) new C();`
Compile error (cannot cast from C to B)

e) `B b = (D) new E();`
No error

f) `A a = (D)((B)new E());`
No error

Question 2.

```java
public abstract class GrandParent {
      public void print() {
            System.out.print("Grandparent ");
            printMe();
            System.out.println();
      }
      protected abstract void printMe();
}
```

```java
public class Parent extends GrandParent {
      @Override
      protected void printMe() {
            System.out.print("Parent ");
      }
}
```

```java
public class Child1 extends Parent {
      @Override
      protected void printMe() {
            super.printMe();
            System.out.print("Child1 ");
     }
}
```

```java
public class Child2 extends Parent {
      @Override
      public void print() {
            System.out.print("Child2 ");
            super.print();
      }
}
```

Given the code for class definitions above, indicate if the code compiles. If it does not compile, show the errors and explain them briefly. If it does compile, what would be the output of the following main method?

```java
public static void main(String[] args) {
      GrandParent[] items = {new Child1(), new Child2()};
      for (GrandParent grandParent : items) {
            grandParent.print();
      }
}
```

It compiles, and the output is:

Grandparent Parent Child1
Child2 Grandparent Parent

Question 3.

Show the output of the following main method.

```
public class Suspicious {

    public static void main(String[] args) {
        try {
            try {
                System.out.print("A");
                bad();
            } catch(Exception e){
                System.out.print("B");
                bad();
            } finally {
                System.out.print("F");
            }
        } catch (Exception e) {
            System.out.print("E");
        }
        System.out.print("G");
    }

    private static void bad() throws Exception {
        try {
            System.out.print("C");
            throw new Exception();
        } finally{
            System.out.print("D");
        }
    }
}
```

Answer:

ACDBCDFEG

Question 4.

```
public class Vehicle {
      public void m1() {
        System.out.println("Vehicle 1");
      }
}
```

```
class Car extends Vehicle {
      public void m1() {
        System.out.println("Car 1 ");
      }

      public void m3() {
        System.out.print("Car 3 ");
        super.m1();
      }
}
```

```
class SportsCar extends Car {
      public void m2() {
        System.out.println("SportsCar 2 ");
      }
}
```

```
class Maserati extends SportsCar {
      public void m2() {
        System.out.print("Maserati 2 ");
        super.m1();
      }
}
```

Considering the variable definitions given below, write down the outputs of each of the Java method calls to the nearby blank areas. Note: Consider each call exists in a separate class file.

```
Object o = new Vehicle();
Vehicle v = new Car();
Car c = new SportsCar();
SportsCar s = new Maserati();
```

```
c.m1();                     Car 1
s.m2();                     Maserati 2 Car 1
c.m3();                     Car 3 Vehicle 1
s.m1();                     Car 1
v.m2();                     Undefined Method
v.m3();                     Undefined Method
c.m2();                     Undefined Method
s.m1();                     Car 1
((Vehicle) o).m1();         Vehicle 1
((Car) o).m1();             ClassCastException
```

Question 5.

What will be the output of the program below?

```
public class MyLists {

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

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

            Collections.reverse(l1);
            l2.remove(0);
            Collections.reverse(l2);
            Collections.reverse(l3);
            l4.remove(0);
            System.out.println(l1);
            System.out.println(l2);
            System.out.println(l3);
            System.out.println(l4);
      }
}
```

Answer:

```
      [1, 3, 5, 7]
      [5, 3]
      [1, 3, 5, 7]
      [5, 3]
```

Question 6.

What will be the output of the program below?

```
import java.util.*;

public class HashMapTest {
   public static void main(String[] args) {
      HashMap<String, ArrayList<String>> multiMap = new HashMap
                                     <String, ArrayList<String>>();

      ArrayList<String> listOne = new ArrayList<String>();
      listOne.add("Blue");
      listOne.add("Black");
      listOne.add("Brown");

      ArrayList<String> listTwo = new ArrayList<String>();
      listTwo.add("Pink");
      listTwo.add("Purple");

      multiMap.put("B color", listOne);
      multiMap.put("P color", listTwo);

      Iterator<String> iteratorMap = multiMap.keySet().iterator();
      System.out.println("The initial size of the map is: " +
                                                  multiMap.size());
      while (iteratorMap.hasNext()) {
         String key = iteratorMap.next();
         System.out.println("Key '" + key + "' has values: " +
                                     multiMap.get(key).toString());
         iteratorMap.remove();
         }
      System.out.println("The final size of the map is: " +
                                                  multiMap.size());
   }
}
```

Answer:

The initial size of the map is: 2
Key 'P color' has values: [Pink, Purple]
Key 'B color' has values: [Blue, Black, Brown]
The final size of the map is: 0

Question 7.

A **stack** is a last in, first out store in main memory, which has *objects* as its elements. It is characterised by two fundamental operations, called **push** and **pop**. The ***push operation*** adds a new object to the top of the stack. If the space allocated to hold the stack is full when the push operation is attempted, then an error condition is raised. The ***pop operation*** removes an object from the top of the stack. A pop reveals previously concealed objects, or results in an empty stack. If the stack is empty when a pop operation is attempted, then an error condition is raised.

Complete the implementation of Java class *MyStack* that implements the stack described above.

```
import java.lang.*;

public class MyStack {

      private Object[] stackArray;
      private int maxSlot;
      private int currentSlot = 0;

      public MyStack (int size) {
            maxSlot = size ;
            stackArray = new Object [size] ;
            }

      public void push(Object o) throws Exception {
            if (currentSlot == maxSlot) {
                  throw new Exception() ;
                  }
            stackArray[currentSlot] = o ;
            currentSlot++ ;
      }

      public Object pop() throws Exception {
            if (currentSlot == 0) {
                  throw new Exception() ;
                  }
            currentSlot-- ;
            return stackArray[currentSlot] ;
            }
      }
```

Question 8.

According to the given code below, answer the questions.

```
public static void main(String[] args) throws Exception{

     RandomAccessFile raf = new RandomAccessFile(new
     File("test.out"),"rw");
     raf.writeBytes("BBM 102");
     raf.writeBytes(" Introduction to Programming");
     raf.seek(8);
     raf.writeBytes("Advanced Programming ");
     raf.writeBytes("Class II ");
     raf.seek(16);
     byte[] bytes = new byte[5];
     raf.read(bytes);
     String str = new String(bytes);
     System.out.println(str);
     long length = raf.length();
     raf.setLength(length + 1);
     raf.seek(raf.length());
     raf.writeBytes("is great!");
     raf.close();
     }
```

a) Write the output of the program.

Prog

b) Write the final contents of *test.out*.

BBM 102 Advanced Programming Class II is great!