BBM102 – Introduction to Programming II – Spring 2017

1st Midterm Exam / 1. Arasınav
April 5, 2017, 1-3 PM

First name / last name: ____

Student ID: _________________________

Section: ⍰ #1 (Dr. Tarhan) / ⍰ #2 (Dr. Akal) / ⍰ #3 (Dr. Ercan) / ⍰ #4 (Dr. Garousi)

Question	1	2	3	4	5	6	7	8	9	10	Total
Points	10	10	10	10	10	10	10	10	10	10	100
Your grade											

Question 1. Write down the output of the following program.

```
public class StringClass {
     public static void main(String[] args) {
            String s = "severe";
            int n = 0;
            for (int i = 0; i < s.length(); i = i+1)
                  if (s.charAt(i) == 'e')
                        n = n * 2;
                  else
                        n = n * 2 + 1;
            System.out.println(n);
            System.out.println(s.concat("-class"));
            if(s.equals("severe-class"))
                  System.out.println("Two strings are equal.");
            else
                  System.out.println("Two strings are not equal.");
      }
}
```

Question 2.

Describe the following concepts with a sentence each

Then, mark the code lines on the program using item labels (such as a, b, …) that correspond to these concepts if they exist in the code. Hint: one of those items have been marked to help you ("//b" for Encapsulation in line 2 of the code).

a) Inheritance:

b) Encapsulation:

c) Overriding:

d) Overloading:

e) Polymorphism:

```
public class Student {
   protected int no;                                // b
   public String name;
   public Student() {no = -1;}
   public void setNo(int pNo) {                     //    b
      System.out.println ("No number is assigned!");
   }
   public int getNo() {                             //    b
      System.out.println ("Number cannot be read!");
      return -1;
   }
}
```

```
public class Undergrad extends Student{             //     a,b
   private String advisor;                          //     b
   public Undergrad() {super();}                    //     a,b
   public void setNo(int pNo) {                     //     a,b,c
      no = pNo;
      System.out.println (no + " is assigned for Undergrad.");
   }
   public int getNo() {                             //     a,b,c
      System.out.println (no + " is read for Undergrad.");
      return no;
   }
}
```

```
public class Graduate extends Student{              //     a,b
   private String supervisor;                       //     b
   public Graduate() {super();}                     //     a,b
   public void setNo(int pNo) {                     //     a,b,c
      no = pNo;
      System.out.println (no + " is assigned for Graduate.");
   }
   public int getNo() {                             //     a,b,c
      System.out.println (no + " is read for Graduate.");
      return no;
   }
}
```

```
public class Main {
     public static void main(String args[]) {
        int myno;
        Student[] students = {new Student(), new Undergrad(), new Graduate()}; //a,e
        students[0].setNo(128);
        students[1].setNo(326);                         //    a,e
        myno = students[0].getNo();
        myno = students[2].getNo();                     //     a,e
     }
}
```

Question 3. Write down the output of the code given in Question2.

```
non static variable cannot be reached

No number is assigned!
326 is assigned for Undergrad.
Number cannot be read!
-1 is read for Undergrad.
```

Question 4. Write down the output of the following program.

```java
public class A {

      public static int x;

      public A(){ x=6;}

      public String toString(){
            return Integer.toString(x);
      }
}

public class B extends A {

      public B(){ x=10;}
}

public class Main{
      public static void main(String[] args){
            A a = new A();
            System.out.print(a);
            a.x=0;
            System.out.print(a);
            B b = new B();
            System.out.print(a);
            System.out.print(b);
            System.out.println(A.x);
            b.x=5;
            System.out.println(b);
            System.out.println(b.x);
      }
}
```

Question 5: Write down the output of the following program

```
public class WhichConstructor {

      WhichConstructor(String s) {
      this(2);
        System.out.println("in string constructor...");
      }

      WhichConstructor(int x) {
            this();
        System.out.println("in int constructor...");
      }

      WhichConstructor() {
        System.out.println("in empty / default constructor...");

      }

      public static void main(String[] args) {
            WhichConstructor wc = new WhichConstructor("test");
      }
}
```

Question 6.

```
public class Animal {
        public void Move() {
          System.out.println("Animal is moving");
        }
}

class Fish extends Animal {
        public void Move() {
           System.out.println("Fish is swimming");
           super.Move();
        }
}

class WalkingMammal extends Animal {
        public void Move() {
          System.out.println("Mammal is walking");
        }
}
```

```
class Dog extends WalkingMammal {
        public void Bark() {
          System.out.println("Dog is barking");
        }

        public void MoveBark() {
          Move();
          Bark();
        }
}

class Husky extends Dog {
        public void Act() {
          System.out.println("Husky is acting");
          super.MoveBark();
        }
}
```

According to the code above, and considering the variable definitions given below, write down the outputs of each of the Java method calls in the nearby blank spaces.

```
Object o = new Animal();
Animal a = new WalkingMammal();
WalkingMammal m = new Dog();
Husky h = new Husky();
Fish f= new Fish();
```

Code	Output
m.Move();	
m.Bark();	
((Dog) m).Bark();	
h.Act();	
f.Move();	
h.Move();	

Question 7: Write down the output of the following program.

```
public class A {
  public A() {
    m1();
  }
  public void m1() {
    System.out.print("a good ");
  }
  public void m2() {
    System.out.print("as a ");
  }
}
```

```
public class B extends A {
  public B() {
    System.out.print("cook ");
  }
}
```

```
public class C extends B {
  public C() {
    System.out.print("as much ");
  }
  public void m1() {
    System.out.print("could ");
  }
  public void m2() {
    System.out.print("cookies as ");
  }
}
```

```
public class D extends B {
  public D() {
    System.out.print("who could ");
    m2();
  }
  public void m1() {
    System.out.print("cookies as ");
    super.m1();
  }
  public void m2() {
    System.out.print("cook cookies ");
  }
}
```

```
public class Twister {
      public static void main(String[] args) {
            A[] as = new A[] {new B(), new C(), new D()};
      }
}
```

Question 8: Write down the output of the following program. Hint: Note that primitive arrays in Java are also Objects.

```
public class Pointy {
      public static void swap(int num1, int num2) {
            int temp = num1;
            num1 = num2;
            num2 = temp;
      }
      public static void swap(int[] values) {
            int temp = values[0];
            values[0] = values[1];
            values[1] = temp;
      }
      public static void main(String[] args) {
            int x=17;
            int y=29;

            swap(x, y);
            System.out.println(x + " - " + y);

            int[] values={17, 29};
            swap(values);
            System.out.println(values[0] + " - " + values[1]);
      }
}
```

Question 9: In computer science, garbage collection (GC) is a form of automatic memory management. The garbage collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program. The *java*.lang.Object.*finalize*() is called by the garbage collector on an object when garbage collection determines that there are no more references to the object. According to these definitions, please put the steps given below in order to show the life time of a Java object.

1. Reclaim Object
2. Create Object
3. Finalize Object
4. Unreachable Object
5. Collect Garbage

Question 10: Design a Card class for a card game. A playing card is defined by a String suit (spades, hearts, diamonds and clubs) and an integer value between 1 to 13. To make sure a created card never changes its suit or value once it is instantiated, your Card class should be immutable (read-only). Make sure your class definition is as restrictive as possible and should make it clear for another developer maintaining your code to understand your immutable design. Your class should have an appropriate equals method. It should be possible to read a Card's value and suit. Complete the following class definition by filling the blanks and adding Constructor, accessors and/or mutators and equals method.

```
public class Card {
      _________________ String suit;
      _________________ int value;









}
```