CPE 203

Final Practice Test

1) Hierarchy

```
public class M_Animal {

    public void greet (M_Animal a){
      System.out.println("Sniff");
    }
}
class Person extends M_Animal {
     public void greet (M_Animal a){
      System.out.println("Grrf");
    }
    public void greet (Person a){
      System.out.println("hi");
    }
}
class Professor extends Person {
     public void greet (Professor a){
      System.out.println("Good Day");
    }
    public void greet (Person a){
      System.out.println("hello");
    }
    public void greet (Student a){
      System.out.println("How can I help you?");
    }
}
class Student extends Person {
     public void greet (Professor a){
      System.out.println("What's the answer?");
    }
    public void greet (Person a){
      System.out.println("hey");
    }
    public void greet (Student a){
      System.out.println("yo");
    }
}
class Driver {
   public static void main(String[] args)    {
      M_Animal rex = new M_Animal();
      M_Animal bob = new Professor();
      Professor sara = new Professor();
      Person jane = new Student();
// What is the output?
      rex.greet(bob);

      bob.greet(rex);
```

```
            sara.greet(bob);

            jane.greet(sara);

            sara.greet(jane);

        }
    }
```

2) **Stream**

```
List <Integer> numbers = Arrays.asList(1,2,3,4,5,6);
int result = 0;
// 1) Convert this for loop to Stream
for (int e: numbers) {
   result += e*2;
}
System.out.println("1) sum: "+result);


// 2)find the double of the first even number that is>3

 List <Integer> numbers2 = Arrays.asList(1,2,3,5,4,6,7,8,9,10);
 int res=0;
 for (int e:numbers2) {
    if (e>3 && e%2==0){
       res = e*2;
       break;    }       }
 System.out.println("3) "+res);
 //Use Stream for problem in part 2
```

3) **Use the class CelestialBody, Planet and the interface Orbits to answer the questions.**

```
public class CelestialBody {
   private double mass;  // in kg
   private double velocity;
   private String name;

   public CelestialBody(double mass, double velocity, String name)
   {
      this.mass = mass;
      this.velocity = velocity;
      this.name = name;
   }
   public double mass() { return mass; }
   public double velocity() { return velocity; }
   public String name() { return name; }
   public boolean equals(Object other) { //code is not given }
}
```

```
public interface Orbits {
   public double duration();
   public CelestialBody orbiting();
}
public class Planet extends CelestialBody implements Orbits {
   private double duration;
   private CelestialBody cb;
   private int moons;

   public Planet(double mass, double velocity, String name,
                 double duration, CelestialBody cb, int moons){
      super(mass, velocity, name);
      this.duration = duration;
      this.cb = cb;
      this.moons = moons;
   }

   public boolean equals(Object other) { //code is not given }
   public double duration() { return duration; }
   public CelestialBody orbiting() { return cb; }
   public int moons() { return moons; }
}
```

In Driver, write a method that accepts a **Map<Planet, List<CelestialBody>>**. Each **Map** entry represents a planet and all celestial bodies that orbit that planet. The method also accepts a **CelestialBody** object and will return a **List** of all the **Planet** objects that are orbited by the **CelestialBody**.

4) Exception

```
/*Exception*/
class TestException {
      public static void m1() {/*throw new ArithmeticException();/* Code
is not given...*/}
      public static void m2() {/*throw new ArithmeticException();/* Code
is not given...*/}
      public static void m3() {/*throw new IllegalArgumentException();/*
Code is not given...*/}
      public static void main(String[] args)    {
         System.out.println("A, ");
         m1();
         try {
            System.out.println("B, ");
            m2();
            System.out.println("C, ");
            m3();
            System.out.println("D, ");
         }
         catch (ArithmeticException e) {
            System.out.println("E, ");
            return;
         }
         catch (Exception e) {
            System.out.println("F, ");
            }
            finally {
               System.out.println("H, ");
            }
            System.out.println("G. ");
         }
    }
```

description	output
Call to m1 and m2 complete normally and, if called, m3 completes normally	
Call to m1 is complete, and m2 throws an ArithmeticException and, if called, m3 complete normally	
Call to m1 and m2 complete normally and, if called, m3 throws IllegalArgumentException	
Call to m1 and m2 complete normally and, if called, m3 throws an ArithmeticException	
Call to m1 throws an ArithmeticException and, if called, others complete normally	