CSE 341 — Java Discussion Questions

1. A static method in Java may not use the `this` reference. Why not?

2. True or false? As soon as an object in Java is no longer accessible from any live variable, its "finalize" method will be invoked and it will be garbage collected.

3. True or false? Java is type safe, but not statically type checked.

4. ```
   class MyPoint {

     public int x, y;

     public MyPoint()
       {this(0,0);}

     public MyPoint(int a)
       {this(a,a);}

     public MyPoint(int x, int y)
       {this.x=x; this.y=y;}

     public void moveTo(int x, int y)
       {this.x = x; this.y = y; }

     public boolean equals (MyPoint p)
       {return (this.x==p.x && this.y==p.y);}

   }
   ```

   What does this print?

   ```
   MyPoint p1 = new MyPoint(10);
   MyPoint p2 = p1;
   MyPoint p3 = new MyPoint(10);

   System.out.println(p1 == p2);
   System.out.println(p1 == p3);
   System.out.println(p1.equals(p2));
   System.out.println(p1.equals(p3));

   p2.moveTo(100,200);
   System.out.println(p1.equals(p2));
   System.out.println(p1.equals(p3));
   ```

5. Consider the following Java class definitions. (These compile correctly.)

   ```
   abstract class Plant {

     // return true if this plant has chlorophyll
     abstract public boolean hasChlorophyll();

     public String description() {
       return "a plant.";
     }
   }
   ```
   ```

```
class Tree extends Plant {
  // height of this tree in feet
  private int height;

  public Tree (int height) {
    this.height = height;
  }

  public String description() {
    return "a tree " + height + " feet tall.  also " + super.description();
  }

  public boolean hasChlorophyll() {
    return true;
  }

  static public String hardness() {
    return "unknown";
  }
}

class Oak extends Tree {

  public Oak (int height) {
    super(height);
  }

  public String description() {
    return "an oak.  also "  + super.description();
  }

  static public String hardness() {
    return "very hard";
  }
}



class Cedar extends Tree {

  public Cedar (int height) {
    super(height);
  }

  public String description() {
    return "a cedar.  also "  + super.description();
  }

  static public String hardness() {
    return "soft";
  }
}
```

```
class Mushroom extends Plant {

  public String description() {
    return "a mushroom.  possibly poisonous, so watch out.  also "
      + super.description();
  }

  public boolean hasChlorophyll() {
    return false;
  }

}
```

Now suppose we also have a class `PlantTest` with a `main` method. What is the result of compiling and executing the Java program for each of the following versions of `PlantTest` and `main`? The result might be that the program runs correctly, or it might have a compile time error, or a runtime error. If the program compiles correctly and can be run, give the output. Otherwise explain what the error is.

(a)

```
public class PlantTest {
  public static void main (String [ ] args) {
    Plant p = new Plant();
    System.out.println(p.hasChlorophyll());
    System.out.println(p.description());
  }
}
```

(b)

```
public class PlantTest {
  public static void main (String [ ] args) {
    Plant p = new Mushroom();
    System.out.println(p.hasChlorophyll());
    System.out.println(p.description());
  }
}
```

(c)

```
public class PlantTest {
  public static void main (String [ ] args) {
    Oak o = new Oak(150);
    Tree t = o;
    System.out.println(o.description());
    System.out.println(t.description());

    System.out.println(o.hardness());
    System.out.println(t.hardness());
  }
}
```

(d)

```
public class PlantTest {
  public static void main (String [ ] args) {
    Oak o = new Oak(150);
    Tree t = (Tree) o;
    System.out.println(t.description());
  }
}
```

(e)

```
public class PlantTest {
  public static void main (String [ ] args) {
    Tree t = new Oak(150);
    Oak o = (Oak) t;
    System.out.println(o.description());
  }
}
```

(f)

```
public class PlantTest {
  public static void main (String [ ] args) {
    Tree t = new Cedar(200);
    Oak o = (Oak) t;
    System.out.println(o.description());
  }
}
```