6. What is the output of running the class C in (a)? What problem arises in compiling the program in (b)?

```
class A {
  public A() {
    System.out.println(
      "A's no-arg constructor is invoked");
  }
}

class B extends A {
}

public class C {
  public static void main(String[] args) {
    B b = new B();
  }
}
```

(a)

```
class A {
  public A(int x) {
  }
}

class B extends A {
  public B() {
  }
}

public class C {
  public static void main(String[] args) {
    B b = new B();
  }
}
```

(b)

A)

Screen

A's no-arg constructor is invoked

B) there is no default constructor in class A

7. Identify the problems in the following code:

```
1  public class Circle {
2    private double radius;
3
4    public Circle(double radius) {
5      radius = radius;
6    }
7
8    public double getRadius() {
9      return radius;
10   }
11
12   public double getArea() {
13     return radius * radius * Math.PI;
14   }
15 }
16
17 class B extends Circle {
18   private double length;
19
20   B(double radius, double length) {
21     Circle(radius);
22     length = length;
23   }
24
25   @Override
26   public double getArea() {
27     return getArea() * length;
28   }
29 }
```

this.radius=radius

No default constructor in A

return super.getArea()*length;

8. If a method in a subclass has the same signature as a method in its superclass with the same return type, is the method overridden or overloaded?

Overridden

9. If a method in a subclass has the same signature as a method in its superclass with a different return type, will this be a problem?

Java supports* covariant return types for overridden methods. This means an overridden method may have a *more* specific return type. That is, as long as the new return type is assignable to the return type of the method you are overriding, it's allowed.