APCS Inheritance Nuances

1.

```
public class Driver
{
    public static void main( String args[] )
    {
        B b = new B( 1234 );
        b.sampleMethod();
    }
}
```

If a subclass's constructor does not include an explicit call to one of its superclass's constructors, then there will be an implicit call to the superclass' default constructor (i.e., the compiler will add a call). If the superclass does not have a default constructor, this implicit call will cause a compile-time error. For example, if we failed to include an explicit call to super in the `ChildrensBook` or `TextBook` constructors, we would get a compile-time error since the `Book` class has no default constructor.

```
public class A
{
    private String name;
    public A()
    {
        System.out.println("A constructor");
        this.assignName();
    }

    public void assignName()
    { name = "Newport"; }

    public String getName()
    { return name; }
}


public class B extends A
{

    private int iD;

    public B( int x )
    { iD = x; }

    public void sampleMethod()
    { System.out.print( this.getName() + " " + iD ); }
}
```

2.

```
public class Demo2
{
    public static void main( String[] args )
    {
        A instance1 = new B();
        B instance2 = (B)instance1;
        System.out.println("instance1.getName() returns ----> " +
            instance1.getName() + "\n" +
                    "instance1.getNumber() returns ----> " +
                    instance1.getNumber() + "\n" +
                    "instance2.getNumber() returns ----> " +
                    instance2.getNumber());
    }
}
```

In addition to overloading methods defined by its superclass, a subclass can also *override* a superclass method; that is, it can define a new version of the method specialized to work on subclass objects. A superclass method is overridden when the subclass defines a method with exactly the same name, the same number of parameters, and the same types of parameters as the superclass.

```
public class A
{
    private String name;
    private int number;

    public A()
    {
        name = "tom";
        number = 1;
    }

    public A( String str )
    {
        name = str;
        number = 1;
    }

    public A( String str, int n )
    {
        name = str;
        number = n;
    }

    public String getName()
    { return name; }

    public int getNumber()
    { return number; }

    public void setNumber( int x )
    { number = x; }

    public void setName( String s )
    { name = s; }
}
```

```
public class B extends A
{
    private double otherNumber;
    public B()
    {
        super();
        otherNumber = 2.5;
    }

    public B( double dob )
    {
        super();
        otherNumber = dob;
    }

    public B( String something,
              int quant, double dob)
    {
        super( something, quant );
        otherNumber = dob;
    }

    public double getOtherNumber()
    { return otherNumber; }

    public int getSumOfNumbers()
    { return (super.getNumber() +
            (int)otherNumber); }

    public int getNumber()
    { return (int)otherNumber; }
}
```

```
public class C extends B
{
    public C()
    { super(); }

    public C( String name,
              double num )
    {
        super( num );
        setName( name );
    }
}
```