1. What is the output of code snippets?

```
class X
{
    //Class X Members
}


class Y
{
    //Class Y Members
}


class Z extends X, Y
{
    //Class Z Members
}
```

Answer: The code will not compile because Java does not support multiple inheritance directly. A class cannot extend more than one class.

2.What is the output of code snippets?

```
class A
{
```

```
    int i = 10;
}

class B extends A
{
    int i = 20;
}

public class MainClass
{
    public static void main(String[] args)
    {
        A a = new B();

        System.out.println(a.i);
    }
}
```

Answer: 10

This is because `a` is of type `A`, and the field `i` of class `A` is accessed, which has the value `10`.

3.What is the output of code snippets?

```
class A
```

```
{
    {
        System.out.println(1);
    }
}

class B extends A
{
    {
        System.out.println(2);
    }
}
class C extends B
{
    {
        System.out.println(3);
    }
}

public class MainClass
{
```

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

Answer:

1

2

3

The instance initializers are executed in the order of inheritance when an instance of `C` is created.

4.What is the output of code snippets?

```
class A
{
    String s = "Class A";
}


class B extends A
{
    String s = "Class B";


    {
```

```
        System.out.println(super.s);
    }
}

class C extends B
{
    String s = "Class C";

    {
        System.out.println(super.s);
    }
}

public class MainClass
{
    public static void main(String[] args)
    {
        C c = new C();

        System.out.println(c.s);
    }
```

}

Answer:

Class A

Class B

Class C

The instance initializers of class `B` and `C` print `super.s`, which refers to the `s` field of their superclass. Finally, `c.s` prints the `s` field of class `C`.

5.What is the output of code snippets?

```
class A
{
   static
   {
      System.out.println("THIRD");
   }
}


class B extends A
{
   static
   {
      System.out.println("SECOND");
```

```
    }
}

class C extends B
{
    static
    {
        System.out.println("FIRST");
    }
}

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

Answer:

THIRD SECOND FIRST

The static initializers are executed in the order of inheritance hierarchy when the class is first loaded.

6.What is the output of code snippets?

```
class A
{
    public A()
    {
        System.out.println("Class A Constructor");
    }
}

class B extends A
{
    public B()
    {
        System.out.println("Class B Constructor");
    }
}

class C extends B
{
    public C()
    {
```

```
        System.out.println("Class C Constructor");
    }
}

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

Answer:

Class A Constructor

Class B Constructor

Class C Constructor

The constructors are called in the order of inheritance hierarchy when an instance of `C` is created.