```
class A {
        A () {
        System.out.println ("Construct A");
        }
}
class B extends A {
        B () {
        System.out.println ("Construct B");
        }
}
class C extends B {
        C () {
        System.out.println ("Construct C");
        }
}
class CallingCons {
        public static void main(String args[]) {
        C c = new C () ;
        }
}
```

Result:
Construct A
Construct B
Construct C

```
class A {
        int i, j;
        A(int a, int b) {
                i = a;
                j = b;
        }
        void show () {
                System.out.println("i  &  j: " + i + " " + j);
        }
}
class B extends A{
        int k;
        B(int a, int b, int c) {
                super (a, b);
                k  = c;
        }
        void show () {
                System.out.println("k: "+ k);
        }
}
class Override {
        public static void main(String args[]) {
        B subOb =  new B(1, 2, 3);
        subOb.show();
        }
}
```

Result:
k: 3

```
class A {
        int i, j;
        A(int a, int b) {
                i = a;
                j  = b;
        }
        void show () {
                System.out.println("i  &  j: " + i + " " + j);
        }
}
class B extends A{
        int k;
        B(int a, int b, int c) {
                super (a, b);
                k = c;
        }
        void show () {
                super.show ();
                System.out.println("k: "+ k);
                }
}
class Override {
        public static void main(String args[]) {
        B subOb = new B(1, 2, 3);
        subOb.show();
        }
}
```

Result:

```
i  &  j: 1 2
k: 3
```

```
class A {
        int i, j;
        A(int a, int b) {
                i = a;
                j  = b;
        }
        void show () {
                System.out.println("i  &  j: " + i + " " + j);
        }
}
class B extends A{
        int k;
        B(int a, int b, int c) {
                super (a, b);
                k  = c;
        }
        void show (String msg) {
                System.out.println(msg + k);
                }
}
class Override {
        public static void main(String args[]) {
        B subOb =  new B(1, 2, 3);
        subOb.show(“This is k “);
        subOb.show();
        }
}
```

Result:

```
This is k 3
i  &  j: 1 2
```

```
// Динамическая диспечеризация методов
class A {
void callMe() {
        System.out.println ("InSide A");
        }
}
class B extends A {
void callMe () {
        System.out.println ("InSide B");
        }
}
class C extends B {
void callMe () {
        System.out.println ("InSide C");
        }
}
class CallingMe {
        public static void main(String args[]) {
        A a = new A ();
        B b = new B ();
        C c = new C () ;
        A r;
        r = a;
        r.callMe();
        r = b;
        r.callMe();
        r = c;
        r.callMe();
        }
}
```

Result:
InSide A
InSide B
InSide C

```
class Figure {
	double dim1;
	double dim2;
	Figure(double a, double b) {
		dim1 = a;
		dim2 = b;
	}
	double area () {
		System.out.println ("Ай донт андеРстенд");
		return 0;
	}
}
class Rectangle extends Figure {
	Rectangle(double a, double b) {
		super (a, b);
	}
	double area () {
		System.out.println("Rectangle ");
		return dim1 * dim2;
	}
}
class Triangle extends Figure {
	Triangle(double a, double b) {
		super (a, b);
	}
	double area () {
		System.out.println("Triangle ");
		return dim1 * dim2 / 2;
	}
}
class FindAreas {
	public static void main(String args[])  {
		Figure f  = new Figure(10, 10);
		Rectangle r = new Rectangle(9, 5);
		Triangle t  = new Triangle(10, 8);
		Figure figref;
		figref = r;
		System.out.println ("Figure = " + figref.area());
		figref = t;
		System.out.println ("Figure = " + figref.area());
		figref = f;
		System.out.println ("Figure = " + figref.area());
	}
}
```

Result:
Rectangle
Figure = 45.0
Triangle
Figure = 40.0
Ай донт андеРстенд
Figure = 0.0