Over loading: We create multiple methods with the same name in the class provided these methods has

different type of arguments or different number of argument

Example 1:

```
public class A {

        public void test(){// 0
                System.out.println("From test");
        }

        public void test(int i){//1
                System.out.println(i);
        }

        public static void main(String[] args) {
                A a1 = new A();
                a1.test();
                a1.test(100);
        }
}
```

Output:

```
From test
100
```

Can we create more than one main method in the same class ?

Example 2:

```
public class A {

	public static void main(String[] args) {//1

		System.out.println("From built in main method");

		A.main();

	}

	public static void main(){// 0

		System.out.println("From user defined method");

	}

}
```

Output:

```
From built in main method

From user defined method
```

Example 3:

```
public class A {

	public static void main(String[] args) {//1

		A a1 = new A();

		a1.emailSender();

		a1.emailSender("avb324");

	}

	public void emailSender(){//0

		System.out.println("Send marketing emailers");

	}

	public void emailSender(String transactionID){//1

		System.out.println("Sending transactional emailer");

	}
}
```

Output:

```
Send marketing emailers

Sending transactional emailer
```

Packages:

1. Packages in java are nothing but folders created to store your programs in organized manner

2. Packages resolves naming convention problems in java, that we can create multiple classes with the

same

3. When you are using a class present in different package then importing would become mandatory

4. When you are accessing the class present in same package then importing it is not required

5. short for importing class is control + shift + o

Example 1:

```
package p1;

public class A {


}
```

Example 2:

```
package p3.p4.p5;


public class C {


}
```

Example 3:

```
package p1;


public class A {
	public int i = 10;
}
package p2;
import p1.A;
public class B {
	public static void main(String[] args) {
		A a1 = new A();
		System.out.println(a1.i);
	}


}
```

Outout:

```
10
```

Example 4:

```
package p1;


public class A {
	public int i = 10;
}
package p2;
```

```
public class B {

        public static void main(String[] args) {

                p1.A a1 = new p1.A();

                System.out.println(a1.i);

        }


}
```

Output:

```
10
```

Example 5:

```
package p1;


public class A {

        public int i = 10;

}

package p1;


public class C {

        public static void main(String[] args) {

                A a1 = new A();

                System.out.println(a1.i);

        }


}
```

Output:

```
10
```

Example 6:

```
package p1;


public class A {

	public int i = 10;

}

package p2;


import p1.A;


public class B extends A{

	public static void main(String[] args) {


	}


}
```

Example 7

```
package p1.p2.p3;


public class D {


}

package p1;


import p1.p2.p3.D;
```

```
public class A {

	public static void main(String[] args) {
		D d1 = new D();
	}
}
```

Output:

Example 8:

```
package p1;

public class A {

	public static void main(String[] args) {

	}
}
package p1;

public class C {
	public static void main(String[] args) {

	}

}
package p2;
```

```
import p1.*;

public class B {

	public static void main(String[] args) {

		A a1 = new A();

		C c1 = new C();


	}


}
```

Example 9:

```
package p1;


public class A {


	public static void main(String[] args) {


	}

}

package p2;

import p1.A;

import p1.p2.p3.D;

public class B {

	public static void main(String[] args) {

		A a1 = new A();

		D d1 = new D();
```

```
            }

}
package p1.p2.p3;

public class D {

}
```

Access Specifier:

Example 1:

```
package p1;

public class A {

            private int i = 10;
            private void test(){
                        System.out.println("From test");
            }
            public static void main(String[] args) {
                        A a1 = new A();
                        System.out.println(a1.i);
                        a1.test();
            }
}
```

Output:

```
10
From test
```

Example 2:

```
package p1;

public class A {

	private int i = 10;
	private void test(){
		System.out.println("From test");
	}

}
package p1;

public class B extends A{
	public static void main(String[] args) {
		B b1 = new B();
		System.out.println(b1.i);
		b1.test();
	}

}
```

Output: Error

Example 3:

```
package p1;


public class A {


	private int i = 10;

	private void test(){

		System.out.println("From test");

	}


}
```

```
package p1;


public class B{

	public static void main(String[] args) {

		A a1 = new A();

		System.out.println(a1.i);

		a1.test();

	}


}
```

Output: Error

Example 4:

```
package p1;


public class A {
        private int i = 10;

        private void test(){

                System.out.println("From test");

        }


}


package p2;

import p1.A;

public class C extends A{

        public static void main(String[] args) {

                C c1 = new C();

                System.out.println(c1.i);

                c1.test();

        }

}
```

Output: Error

Example 5:

```
package p1;


public class A {


        private int i = 10;

        private void test(){

                System.out.println("From test");

        }
}


package p2;

import p1.A;

public class C{

	public static void main(String[] args) {

		A a1 = new A();

		System.out.println(a1.i);

		a1.test();

	}

}
```

Output: Error

Examle 6:

```
package p1;


public class A {


	int i = 10;

	void test(){

		System.out.println("From test");

	}

	public static void main(String[] args) {

		A a1 = new A();

		System.out.println(a1.i);
```

```
                a1.test();
        }

}
```

Example 7:

```
package p1;

public class A {

        int i = 10;
        void test(){
                System.out.println("From test");
        }

}
package p1;

public class B extends A{
        public static void main(String[] args) {
                B b1 = new B();
                System.out.println(b1.i);
                b1.test();
        }

}
```

Output:

```
10
From test
```

Example 8:

```
package p1;

public class A {

	int i = 10;
	void test(){
		System.out.println("From test");
	}


}
package p1;

public class B{
	public static void main(String[] args) {
		A a1 = new A();
		System.out.println(a1.i);
		a1.test();
	}

}
```

Output:

```
10
```

```
From test
```

Example 9:

```java
package p1;

public class A {

	int i = 10;
	void test(){
		System.out.println("From test");
	}

}
package p2;
import p1.A;
public class C extends A{
	public static void main(String[] args) {
		C c1 = new C();
		System.out.println(c1.i);
		c1.test();
	}
}
```

Output: Error

Example 10:

```java
package p1;
```

```
public class A {


	int i = 10;

	void test(){

		System.out.println("From test");

	}




}

package p2;

import p1.A;

public class C {

	public static void main(String[] args) {

		A a1 = new A();

		System.out.println(a1.i);

		a1.test();

	}

}
```

Output: Error

Note:

a. If you make your class member as private then those members can be accessed only in same class

b. If you make your class member as default then those members can be accessed only in same package

c. If you make your class member as protected then those members can be accessed in same package and

different package only through inheritance

d. If you make your class member as public then those members can be accessed every where

Protected Access Specifier:

Example 1:

```
package p1;

public class A {

	protected int i = 10;
	protected void test(){
		System.out.println("From test");
	}
	public static void main(String[] args) {
		A a1 = new A();
		System.out.println(a1.i);
		a1.test();
	}

}
```

Outout:

```
10
From test
```

Example 2:

```
package p1;
```

```
public class A {


	protected int i = 10;

	protected void test(){

		System.out.println("From test");

	}


}
```

```
package p1;


public class B extends A{


	public static void main(String[] args) {

		B b1 = new B();

		System.out.println(b1.i);

		b1.test();

	}


}
```

Outout:

```
10

From test
```

Example 3:

```
package p1;


public class A {
        protected int i = 10;

        protected void test(){

                System.out.println("From test");

        }


}

package p1;


public class B{


        public static void main(String[] args) {

                A a1 = new A();

                System.out.println(a1.i);

                a1.test();

        }


}
```

Output:

```
10

From test
```

Example 4:

```
package p1;


public class A {
        protected int i = 10;

        protected void test(){

                System.out.println("From test");

        }


package p2;

import p1.A;


public class C extends A{

        public static void main(String[] args) {

                C c1 = new C();

                System.out.println(c1.i);

                c1.test();

        }

}

}
```

Output:

```
10

From test
```

public access specifier:

Example 1:

```
package p1;


public class A {
        public int i = 10;

        public void test(){

                System.out.println("From test");

        }

        public static void main(String[] args) {

                A a1 = new A();

                System.out.println(a1.i);

                a1.test();

        }


}
```

Output:

```
10

From test
```

Example 2:

```
package p1;


public class A {


        public int i = 10;

        public void test(){

                System.out.println("From test");

        }


}
```

```
package p1;

public class B extends A{

	public static void main(String[] args) {
		B b1 = new B();
		System.out.println(b1.i);
		b1.test();
	}

}
```

Output:

```
10
From test
```

Example 3:

```
package p1;

public class A {

	public int i = 10;
	public void test(){
		System.out.println("From test");
	}

}
```

```
package p1;

public class B{

	public static void main(String[] args) {
		A a1 = new A();
		System.out.println(a1.i);
		a1.test();
	}

}
```

Output:

```
10
From test
```

Access Specifier a class supports:

1. public- A public class can be accessed in any packages
2. default- A default class can be accessed only in the same package

private and protected a class would not support

Example 1: for default class

```
package p1;

class A {// This class can be used only in same package
```

```
	public int i = 10;

	public void test(){

		System.out.println("From test");

	}


}


package p1;


public class B{


	public static void main(String[] args) {

		A a1 = new A();

		System.out.println(a1.i);

		a1.test();

	}


}


package p2;


import p1.A;//Error



public class C {

	public static void main(String[] args) {

		A a1 = new A();//Error
```

```
                System.out.println(a1.i);
                a1.test();
        }
}
```

Example 2: An example for public class

```
package p1;

public class A {// This class can be used only in same package

        public int i = 10;
        public void test(){
                System.out.println("From test");
        }

}

package p1;

public class B{

        public static void main(String[] args) {
                A a1 = new A();
                System.out.println(a1.i);
                a1.test();
        }
}

package p2;


import p1.A;



public class C {

	public static void main(String[] args) {

		A a1 = new A();//p1

		System.out.println(a1.i);

		a1.test();

	}

}
```

What access specifiers constructors would support ?

a. constructor can be priavte and for such contructors object should be created in same class

b. constructor can be default and for such contructors object should be created any where in same

package

c. constructor can be protected and for such contructors object should be created any where in same

package which is very similar to default constructors

d. constructor can be public and for such contructors object can be created any where in the program

Example 1:

```
package p1;

public class A {

   private          A(){
                    System.out.println("From Constructor A");
         }

   public static void main(String[] args) {
                    A a1 = new A();
         }

}
```

Output:

```
From Constructor A
```

Example 2:

```
package p1;

public class A {

   private          A(){
                    System.out.println("From Constructor A");
         }
}

package p2;


import p1.A;


public class B {


        public static void main(String[] args) {

                A a1 = new A();

            }




}
```

Output:

```
Error
```

Example 3:

```
package p1;


public class A {

  private       A(){

                System.out.println("From Constructor A");

         }

}

package p1;


public class C {
```

```
                public static void main(String[] args) {
                                A a1 = new A();
                        }
}
```

Output:

```
Error
```

Example 4:

```
package p1;

public class A {
   A(){
                        System.out.println("From Constructor A");
                }
   public static void main(String[] args) {
                        A a1 = new A();
                }
}
```

Output:

```
From Constructor A
```

Example 5:

```
package p1;

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


}
package p1;


public class C {
               public static void main(String[] args) {
                                          A a1 = new A();
          }
}
```

Output:

```
From Constructor A
```

Example 6:

```
package p1;


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


}
package p2;


import p1.A;
```

```
public class B {

        public static void main(String[] args) {
                A a1 = new A();
            }

}
```

Output:

Error

Example 7:

```
package p1;

public class A {
        protected A() {
                System.out.println("From Constructor A");
        }

        public static void main(String[] args) {
                A a1 = new A();

        }

}
```

Output:

```
From Constructor A
```

Example 8:

```
package p1;


public class A {
	protected A() {
		System.out.println("From Constructor A");
	}



}
```

```
package p1;


public class C {
	public static void main(String[] args) {
		A a1 = new A();
	}
}
```

Output:

```
From Constructor A
```

Example 9:

```
package p1;


public class A {
```

```
          protected A() {

                    System.out.println("From Constructor A");

          }




}

package p2;


import p1.A;


public class B {


           public static void main(String[] args) {

                              A a1 = new A();

                    }




}
```

Output:

```
Error
```

Example 10:

```
package p1;

public class A {

          public A() {

                    System.out.println("From Constructor A");

          }
```

```
	public static void main(String[] args) {

		A a1 = new A();

	}

}
```

Output:

```
From Constructor A
```

Example 11:

```
package p1;

public class A {

	public A() {

		System.out.println("From Constructor A");

	}


}
```

```
package p1;


public class C {

	public static void main(String[] args) {

		A a1 = new A();

	}

}
```

Output:

```
From Constructor A
```

Example 12:

```
package p1;
```

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


}
package p2;
import p1.A;
public class B extends A{


	 public static void main(String[] args) {
			A a1 = new A();
		}
}
```

Output:

```
From Constructor A
```

Polymorphism:

Interview Questions

note:

a. During overriding accessspecifiers need not be same

b. During overriding the scope of accessspecifier should not be reduced

Question 1:

```
package p1;
public class A {
	protected void test(){
		System.out.println(100);

	}


}

package p1;

public class B extends A{

	@Override

	void test(){

		System.out.println(500);

	}

	public static void main(String[] args) {

		B b1 = new B();

		b1.test();

	}


}
```

Output: Error

Question 2:

```
package p1;

public class A {

	void test(){

		System.out.println(100);

	}


}

package p1;
```

```
public class B extends A{
	@Override
	protected void test(){
		System.out.println(500);
	}
	public static void main(String[] args) {
		B b1 = new B();
		b1.test();
	}
}
```

Output:

```
500
```

Question 3:

```
package p1;
public class A {
	public void test(){
		System.out.println(100);
	}

}
package p1;
public class B extends A{
	@Override
	protected void test(){
		System.out.println(500);
	}
```

```
    public static void main(String[] args) {
        B b1 = new B();
        b1.test();
    }
}
```

Output:

```
Error
```

Super keyword:

a. It helps us to access members of parent class. super keyword can be used only when inheritance is

happening

b. super keyword nnot be used inside static methods.

c. We cannot use super keyword in main method because main method is static

d. using super keyword we can call construtors of parent class. but ensure that to call parent class

constructor you are using super keyword in child class constructor

e. super keyword cannot be second statement while calling parent class constructor from child class

constructor.

ex: super();

Example 1:

```
package p1;
public class A {
    int i = 10;
    public void test(){
        System.out.println(100);
    }
}
```

```
package p1;
public class B extends A{

    public static void main(String[] args) {
        B b1 = new B();
        b1.x();
    }
    public void x(){
        System.out.println(super.i);
        super.test();
    }
}
```

Output:

```
10
100
```

Example 2:

```
package p1;
public class A {
    int i = 10;
    public void test(){
        System.out.println(100);
    }
}

package p1;
public class B extends A{

	public static void main(String[] args) {
		B b1 = new B();
		b1.x();
	}
	public static void x(){
		System.out.println(super.i);//Error
		super.test();//Error
	}
}
```

Output:

```
Error
```

Example 3:

```
package p1;
public class A {
	static int i = 10;
	public static void test(){
		System.out.println(100);
	}
}
package p1;
public class B extends A{
        public static void main(String[] args) {
                B b1 = new B();
                b1.x();
        }
        public void x(){
                System.out.println(super.i);
                super.test();
        }
}
```

Output:

```
10
100
```

Example 4:

```
package p1;
public class A {
        A(){
                System.out.println("From Constructor A");
        }
}

package p1;
public class B extends A{

        B(){
                super();
```

```
	}

	public static void main(String[] args) {
		B b1 = new B();

	}

}
```

Output:

```
From Constructor A
```

Example 5:

```
package p1;
public class A {
	A(int i){
		System.out.println(i);
	}
}

package p1;
public class B extends A{

	B(){
		super(500);
	}

	public static void main(String[] args) {
		B b1 = new B();

	}

}
```

Output:

```
500
```

Example 6:

```
package p1;
public class A {
	A(int i){
		System.out.println(i);
	}
}

package p1;
public class B extends A{

	B(){
		System.out.println("From constructor B");
		super(500);
	}

	public static void main(String[] args) {
		B b1 = new B();
```

```
        }


}
```

Output:

Error

Example 7:

```
package p1;

public class A {

        A(int i){

                System.out.println(i);

        }

}

package p1;

public class B extends A{


        B(){

                super(500);

                System.out.println("From constructor B");


        }


        public static void main(String[] args) {

                B b1 = new B();


        }
}
```

Output:

```
500

From constructor B
```