packages:-

--there are 3 uses of the packages in java:

1.with the package,we can bind the related concept logically.

com.masai.db :
100: --- with intract with the DB

com.masai.service
200: -- in which we define business logic to process the data

com.masai.presentation
200: -- in which we define the presentation logic.

--in java api we have so many predefined classes and interfaces are there.
they are also categoriesed in diff packages.

java.lang : it contains all the common classes(ex: String, System, Object, etc)

java.io package : contains all the classes and interface which are required to perform i/o operations

java.sql package

java.util package :

java.net package:

2. with the help of package we can avoid the naming conflict.

String
java.lang.String --- fully qualified name

com.masai.String ---

3. to provide some level of security, by using access modifiers.

default access modifier : outside the package we can not access

protected access modifier: outside the package we can access using inheritance.

--Note: in realtime we should not define a class, interface without a package.

--In Java every package is a folder, where as every folder is not a package.

--when we keep any classes inside any package, then we can compile that class by following
2 ways:

1. manually: here first we need to compile the class and then we need to create folder with the
name of the package and keep all the .class files inside that folder. after that we can run our application by using fully qualified name.

>java p1.Demo

2.automatically: here we need to compile our class by using following command:

>javac -d . Demo.java :- this command wll create a folder with the name of packages automatically.

Abstract class:
==========

abstract class
abstract method

--by using an Abstract class we acheive partial abstraction in java.

abstract Animal

eat()
sleep()
walk()
makeNoise()

DomesticAnimal WildAnimal

Dog Cat Tiger Lion

Dog tommy = new Dog();

Animal tommy = new Dog();

Animal tommy = new Animal(); //CE

--In java, if we want to restrict the user, so that he can not instantiate or create object of a class directly then marks that class as abstract class.

--if we make constructor as private then we can not create its object outside that class but inside the class we can create its object

Note: an abstract class does not have any meaning, unless it is extended by the child class, and we are allowed to create only child class object.

there are 3 differences between a normal class and an abstract class:
==========

1. for an abstract class we can not create its object directly,(new keyword is not applicable)

2.inside an abstract class, we **may have an abstract method also.(it is not mandatory)
Note: we can have an empty abstract class also.

3. final keyword is not applicable with the abstract class (final and abstract both are enimies)

Note: apart from above 3 diff bt normal and abstract class, we can do all the things with the abstract class whatever we can do with the normal class.

ex:

we can have variables
we can have constructors.
etc..

abstract method:

--method with body is also knwon as implemented method or concreate method

--method without body is known as abstract method or unimplemented method. these types of methos should have abstract keyword.

ex:

```
public abstract void makeNoise();
```

Note: - inside a normal class/concreate class, we can not have an abstract method.

--if we want to place an abstract method inside our class then we need to mark that class also as an abstract class.

ex:

A.java:

```
package com.masai;

public abstract class A {

	//unimplemented method or abstract method
	 public abstract void funAbs();


	//implemented method
	void funA() {
```

```
		System.out.println("inside funA of A");
	}


}
```

rule:
*****Note: if an abstract class having any abstract method then we must override that method inside the child class otherwise we need to mark that child class also as an abstract class.**

A.java:

```
package com.masai;

public abstract class A{



	//unimplemented method or abstract method
	 public abstract void funAbs1();

	//unimplemented method or abstract method
		public abstract void funAbs2();

		//unimplemented method or abstract method
		 public abstract void funAbs3();



	//implemented method, inside child class we can not override this
	 final void funA() {

		System.out.println("inside funA of A");
	}


}
```

AChild.java:

```
package com.masai;

public abstract class AChild extends A{

	@Override
	public void funAbs1() {
		// TODO Auto-generated method stub

	}

	@Override
	public void funAbs2() {
		// TODO Auto-generated method stub

	}

}
```

AGrandChild.java:

```
package com.masai;

public class AGrandChild extends AChild{

	@Override
	public void funAbs3() {
		// TODO Auto-generated method stub

	}

}
```

Normal class

```
A a1 = ?  // 3 possible values

A a1 = new A(); // same class obj
A a1 = new AChild();//  child class obj
A a1=null;
```

Abstract class:

Abs a1 =? // 2 possible values

Abs a1 = new Abs();// CE
Abs a1 = new AbsChild();
Abs a1 = null;

Abstract object created along with its child class object.

example:

A.java:

```
package com.masai;

public abstract class A{


	public A() {
		System.out.println("inside constructor of A ");
	}


	//unimplemented method or abstract method
	 public abstract void funAbs1();


	//implemented method
	 void funA() {

		System.out.println("inside funA of A");
	}


}
```

AChild.java:

```
----------

package com.masai;

public class AChild extends A{


	public AChild() {
		System.out.println("inside the constructor of AChild");
	}



	@Override
	public void funAbs1() {

		System.out.println("inside funAbs() of AChild");

	}
}
```

Demo.java:

```
----------

package com.masai;

public class Demo {


	public static void main(String[] args) {

		A a1 = new AChild();

		a1.funA();
		a1.funAbs1();


	}

}
```

*****Note: an abstract method can not be static.**

In Java we have 3 types of valid structure:

1. full 100% implemented structure (concrete class) // implemented means, method with body

2.partial implemented structure (abstract class) // partial abstraction

3.full 100% unimplemented structure (interface) // 100%