Object Oriented Programming
CS F213

J. Jennifer Ranjani

email: jennifer.ranjani@pilani.bits-pilani.ac.in

Chamber: 6121 B, NAB

Consultation: Appointment by e-mail

BITS Pilani

Pilani Campus

-Interfaces
-Nested Classes

BITS Pilani
Pilani Campus

Queries asked in the previous class

- What is the output?

```
class Parent{
static int A=50;
static void show() {
System.out.println(A);
}}

class Child extends Parent{
int A=10;
int show() {
System.out.println(A);
}}
```

Compilation Error:
Instance method cannot override a static method from parent

Queries asked in the previous class

```
class Parent{
static int A=50;
static void show() {
System.out.println(A);
}}

class Child extends Parent{
int A=10;
}
class test{
public static void main(String args[]) {
Child c = new Child();
c.show();
System.out.println(c.A);    }
}
```

Output:
50
10

Warning:
The static method show() from the type Parent should be accessed in a static way

Queries asked in the previous class


```
interface Printable{
void show();
}

interface Showable{
int show();
}

class trial implements Printable,Showable{
public int show() {
System.out.println("Within Show");
}
}
```

Error:
Return type is incompatible

Queries asked in the previous class

```
interface Printable{
static void show() {
System.out.println("Within Static
    Show");
};
}

interface Showable{
default void show()
{
System.out.println("Within default
    Show");
};
}

class trial implements
    Printable,Showable{
}
class test{
public static void main(String
    args[]) {
trial t = new trial();
t.show();
}
}
```

Output:
Within default Show

Queries asked in the previous class

```
interface Printable{
static void show() {
System.out.println("Within Static
    Show");
};
}

interface Showable{
default void show()
{
System.out.println("Within default
    Show");
};
}
```

```
class trial implements
    Printable,Showable{
public void show() {
System.out.println("Within Show");
Showable.super.show();
Printable.show();
} }
class test{
public static void main(String
    args[]) {
trial t = new trial();
t.show();
}
}
```

Output:
Within Show
Within default Show
Within Static Show

Queries asked in the previous class

```
interface Printable{
int data=20;
class Showable{
void show()
{
System.out.println("Interface Variable "+data);
}
}
}
class test extends Printable.Showable{
public static void main(String args[]) {
test c = new test();
c.show();
}}
```

Output:
Interface Variable 20

innovate achieve lead

BITS Pilani

Pilani Campus

Nested Classes

Inner Classes

- Nested classes are used to logically group classes or interfaces in one place, for more readability and maintainability.
- Nested class can access all members of the outer class including the private data members and methods.
- Two types:
  - Non-static nested class (inner class)
    - Member
    - Anonymous
    - Local
  - Static nested class

Member Inner class - Example

innovate achieve lead

```
class Outer{
int data = 30;
private int val = 20;
class Inner{
void show() {
System.out.println("Data=:"+data+"Value="+val);} }
}
class test{
public static void main(String args[]) {
Outer o = new Outer();
Outer.Inner in = o.new Inner();
in.show();
} }
```

Member Inner Class

- Compiler creates two class files of the inner class
  - Outer.class and Outer$Inner.class
- To instantiate the inner class, the instance of the outer class must be created
- The inner class have a reference to the outer class, thus it can access all the data members of the outer class.

Member Inner class - Example

innovate achieve lead

```
class Outer{
int data = 30;
private int val = 20;
private class Inner{
void show() {
System.out.println("Data=:"+data+"Value="+val);}
}
void print() {
Inner in = new Inner();
in.show();}
}
class test{
public static void main(String args[]) {
o.print();}
}
```

Note:
Unlike a class, an inner class can be private.

Anonymous Inner Class

- Class with no name
- Used when a method or interface is to be overridden

Anonymous class - Example

```
abstract class Outer{
int data = 30;
abstract void show();
void print() {
System.out.println("Within Print");
}
}
class test {
public static void main(String args[]) {
Outer o = new Outer() {
void show() {
System.out.println("Data=:"+data);}
};
o.show();
o.print();
}}
```

Anonymous Class

- The name of the class created is decided by the compiler
- In the given example, the anonymous class extends the ‘Outer’ class and gives implementation for the show() method.
- The object of the anonymous class can be referred by the reference variable ‘o’
- Anonymous class cannot have additional methods because it is accessed using the reference to the ‘Outer’ class

Anonymous Inner Class using Interface-Example

innovate achieve lead

```
interface Outer{
int data = 30;
void show();
}
class test {
public static void main(String args[]) {
Outer o = new Outer() {
public void show() {
System.out.println("Data=:"+data);
//data =25;          // Error
}
};
o.show();
}
}
```

Local Inner Class-Example

```
class Outer{
private int data = 30;
void show() {
int val =50;
class inner{
void print() {
System.out.println("Value= "+val+"Data="+data);}}
inner i =new inner();      // Creating a named type
i.print(); }
}
class test {
public static void main(String args[]) {
Outer o = new Outer();
o.show();
//o.print();  //Error}
}
```

Note:
Local inner class can be instantiated only within the method it is defined.

Static Inner Class-Example

```
class Outer{
static int data = 30;
private static int val = 20;
static class Inner{
void show() {
System.out.println("Data=:"+data+"Value="+val);
}
}
}
class test{
public static void main(String args[]) {
Outer.Inner in = new Outer.Inner();
in.show();
}
}
```

Static Inner Class

- A static class created inside a class.
- It can access the static data members of the outer class including the private members.
- It cannot access the non-static members and methods.
- The object of the outer class need not be created, because static methods or classes can be accessed without object.
- **Note:** Only inner classes can be prefixed with the static keyword.

Take Home Exercise: What happens when the Interface Showable is private? Update the following code.


```
class Printable{
public void print()
{
System.out.println("Within Print");
}
interface Showable{
void show();}
}

class trial extends Printable
    implements Printable.Showable
    {
public void show()
{
System.out.println("Within Show");
} }
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.show();
t.print();
}
}
```

Output:
Within Show
Within Print

innovate achieve lead

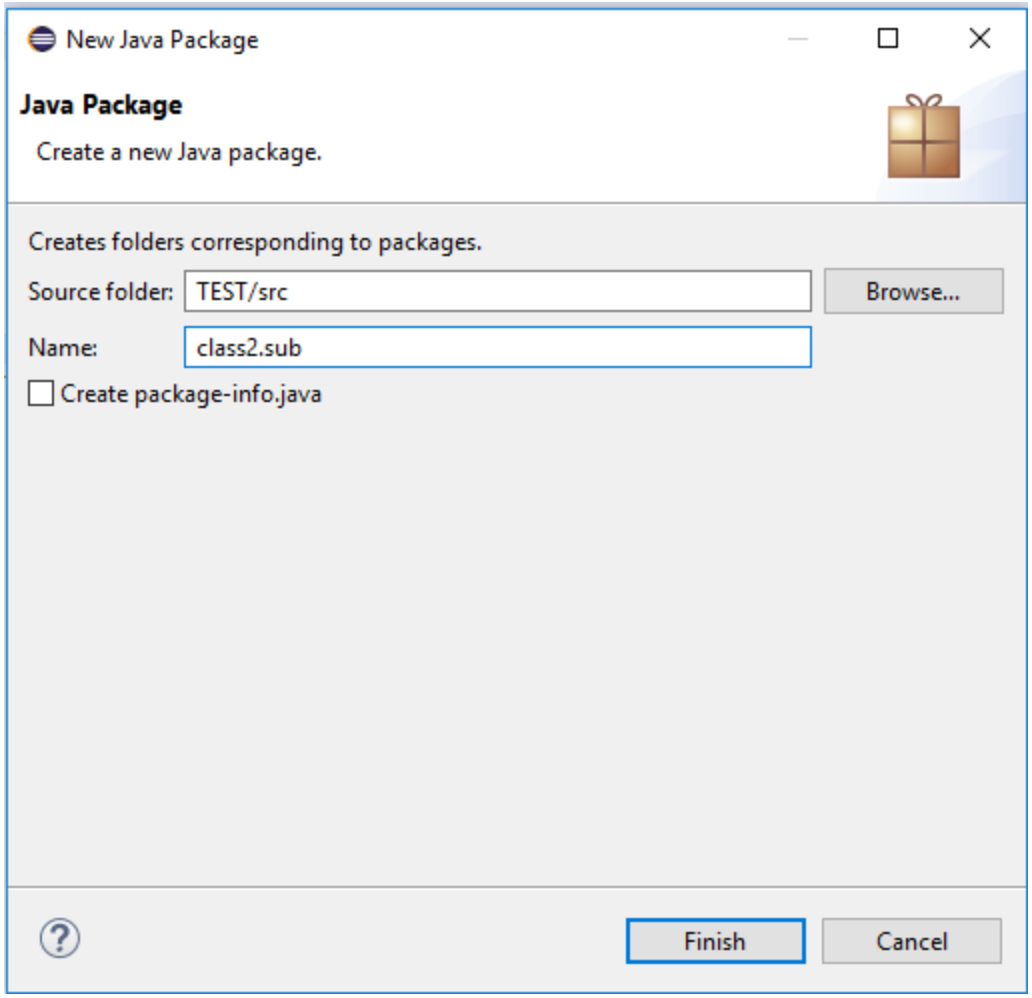
BITS Pilani

Pilani Campus

Packages

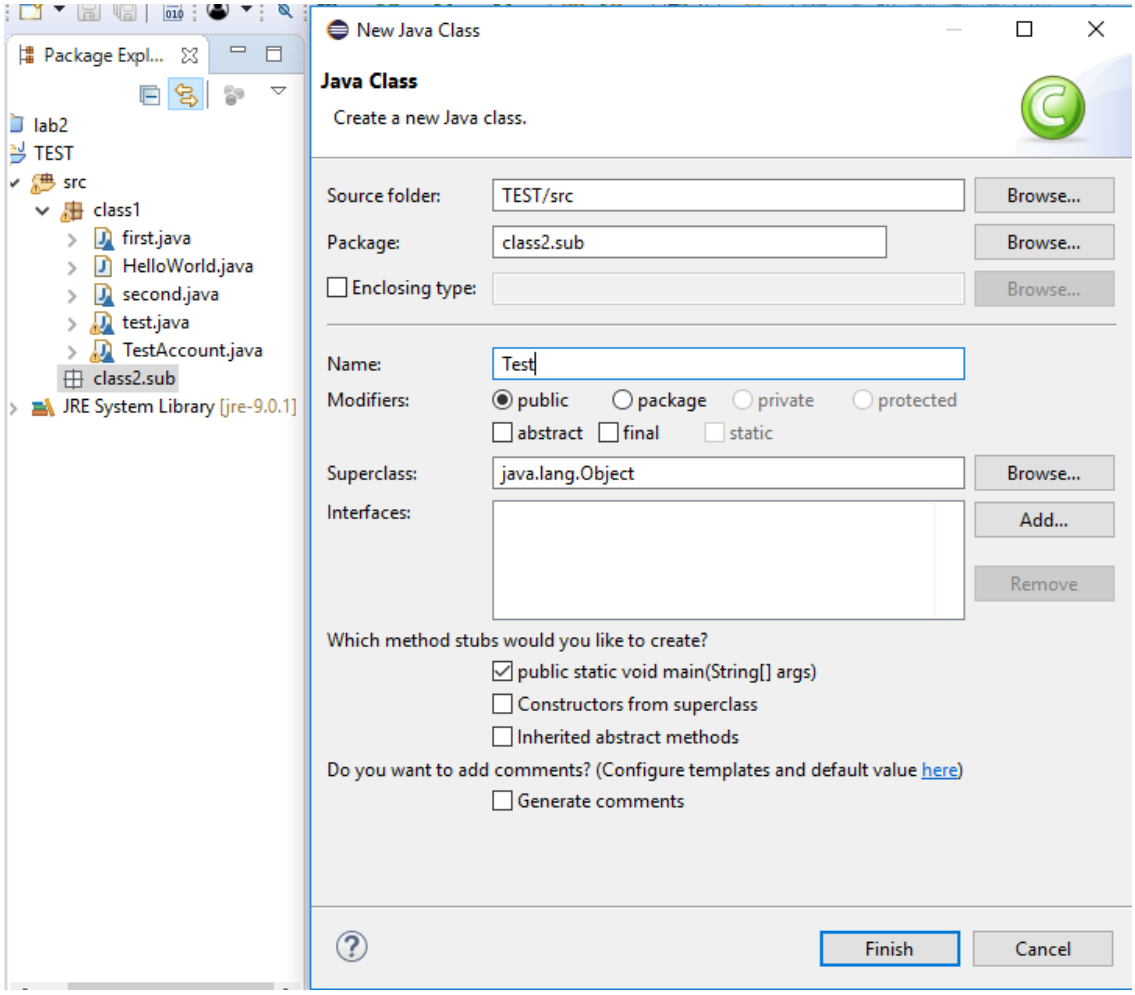
Create a package & sub package

Project → New → Package



Create a class within the package

Package → New → class



Class within the package

```
package class2.sub;

public class Test {

public static void main(String[] args) {
// TODO Auto-generated method stub

}

}
```

Importing a package

```
package class1;

public class HelloWorld
{
 public void show() {
  System.out.println("Within class
     1's show");
  }
}
```

```
package class2.sub;
import class1.*;

public class Test {

public static void main(String[]
     args) {
HelloWorld h = new HelloWorld();
h.show();

}

}
```

Importing a class

```
package class1;

public class HelloWorld
{
 public void show() {
  System.out.println("Within class
     1's show");
  }
}
```

```
package class2.sub;
import class1.HelloWorld;

public class Test {
public static void main(String[]
     args) {
HelloWorld h = new HelloWorld();
h.show();
}
}
```

Take Home Exercise: Learn how to execute the same code from the command prompt.

Access Modifiers

innovate achieve lead

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y