

## Lesson 03 Demo 04

### Implementing types of Inheritance

**Objective:** To implement the inheritance types

**Tools:** Eclipse IDE

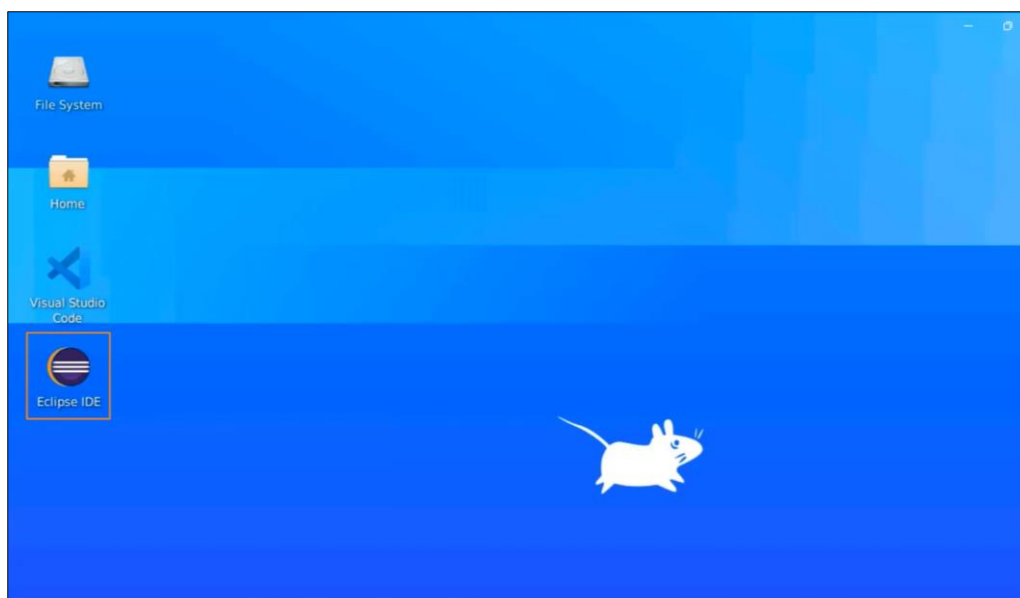
**Prerequisites:** None

Steps to be followed:

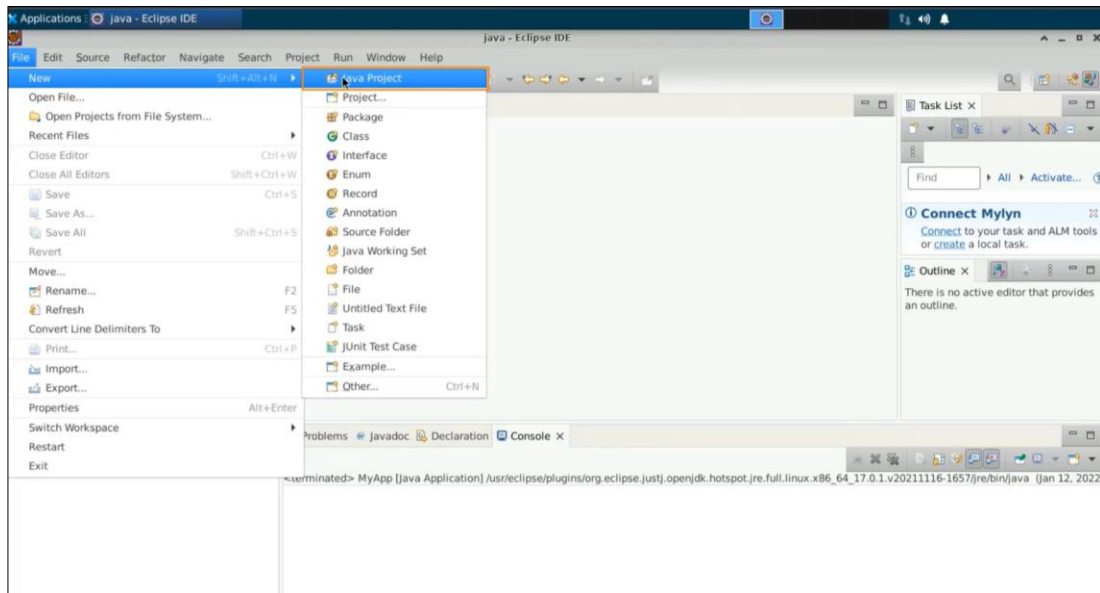
1. Open Eclipse IDE, and a new Java project and a class
2. Implement single-level inheritance
3. Implement multi-level inheritance
4. Understand the concept of hierarchy with example data
5. Implement multiple inheritances with example data
6. Understand and use the concept of hybrid

#### Step 1: Open Eclipse IDE, and a new Java project and a class

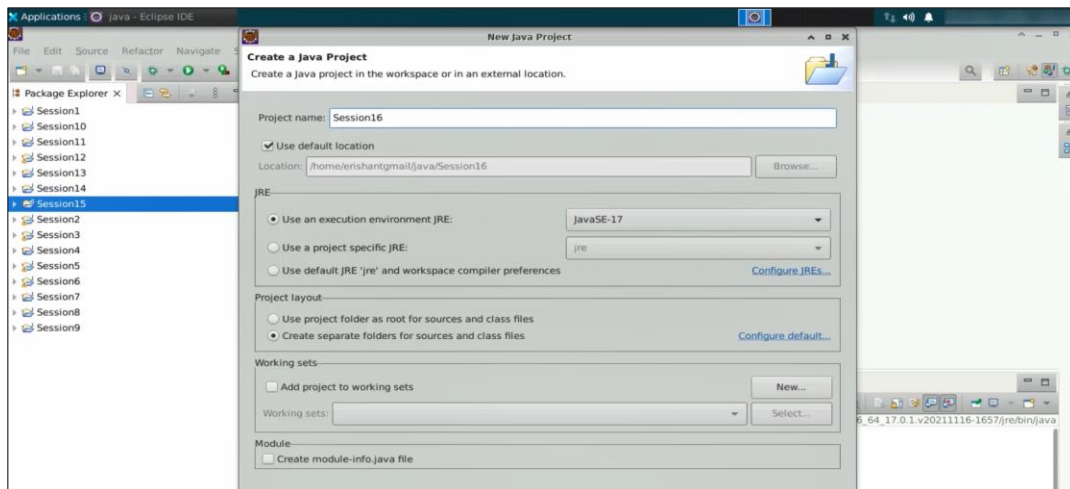
##### 1.1 Open the Eclipse IDE



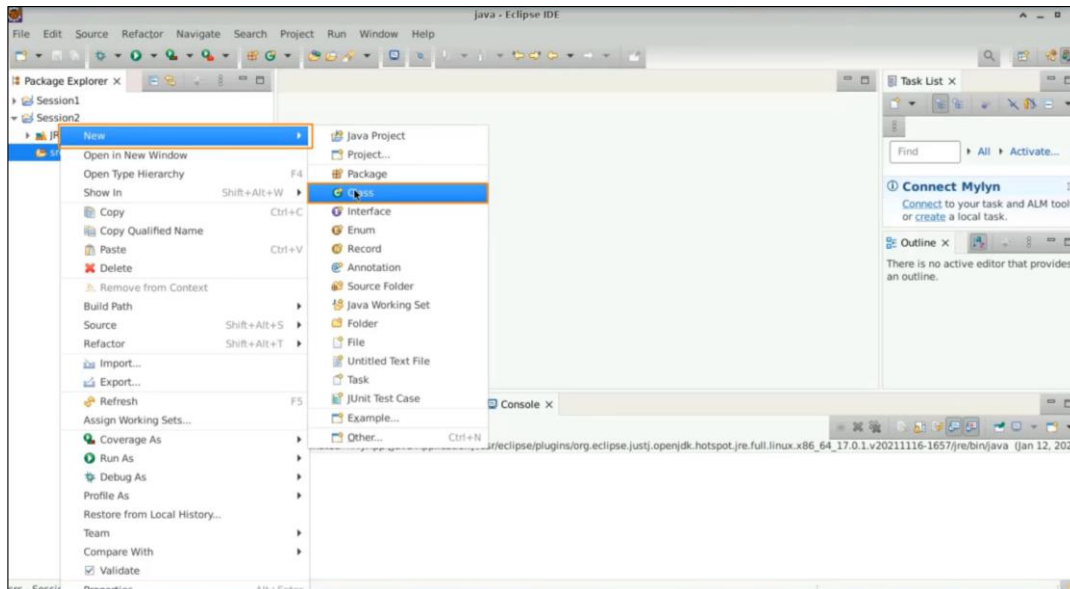
## 1.2. Select **File**, then **New**, and then **Java project**



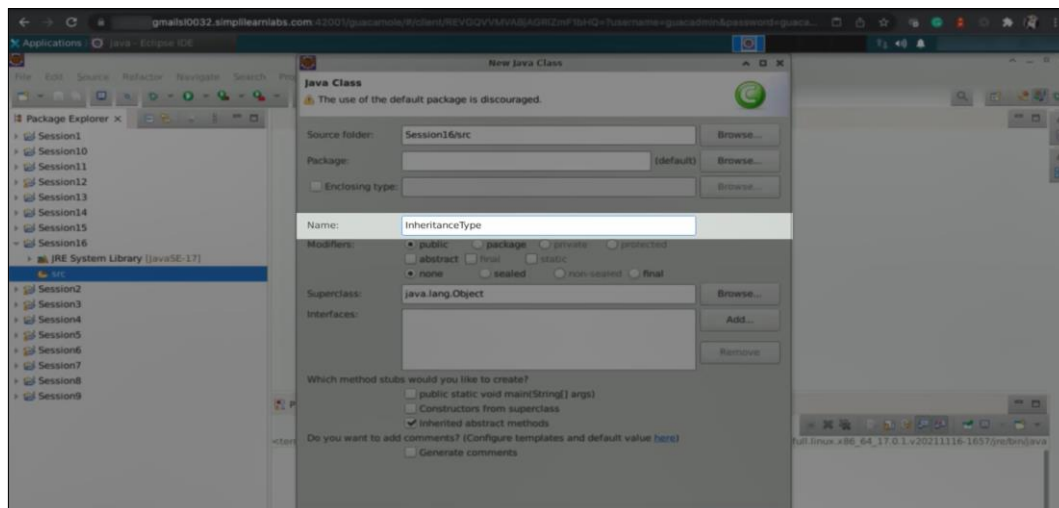
## 1.3 Name the project **“Session16”**, uncheck **“Create a module info dot Java file”**, and press **Finish**



1.4 With a **Session16** on the src, do a right-click and create a **new class**

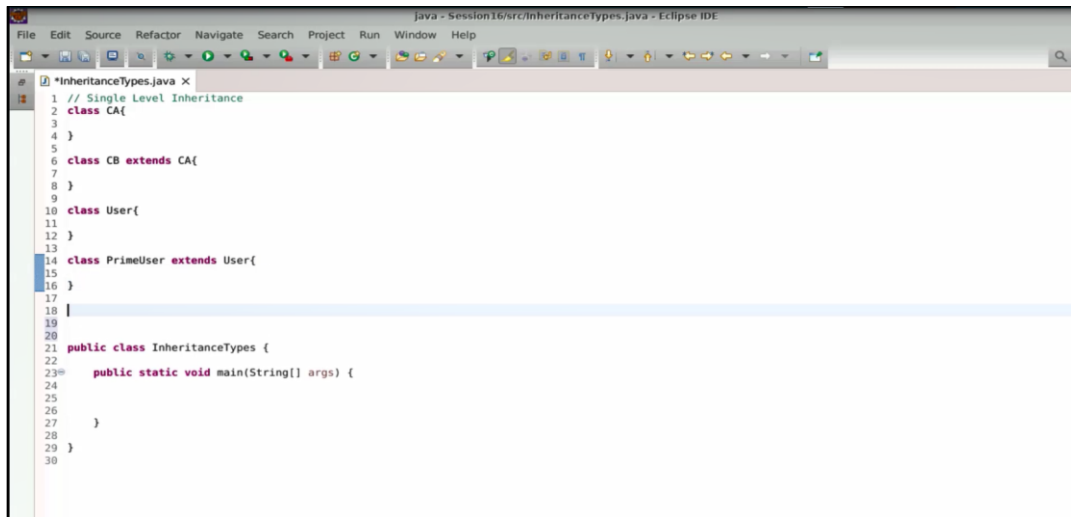


1.5 Name this class as an **InheritanceTypes**, then select the **main method**, and then select **finish**



## Step 2: Implement single-level inheritance

- 2.1 There's a class called CA and a class CV, which is going to extend CA. This technique is known as **single-level inheritance**. Where what you have are a parent and one child, that is what is a single-level inheritance. what can be an example of this single-level inheritance, there is a class called user and there can be a class called Prime User which extends the user



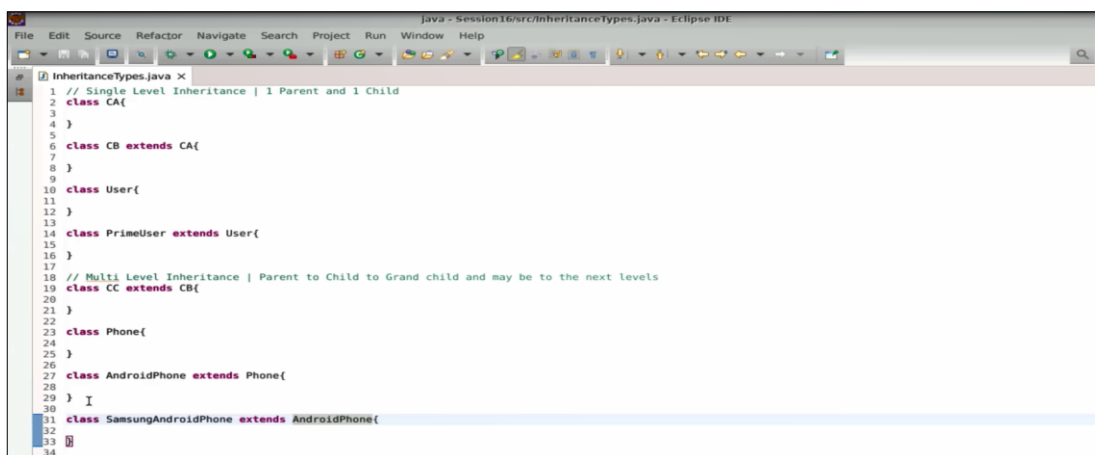
```

1 // Single Level Inheritance
2 class CA{
3
4 }
5
6 class CB extends CA{
7
8 }
9
10 class User{
11
12 }
13
14 class PrimeUser extends User{
15
16 }
17
18
19
20
21 public class InheritanceTypes {
22
23     public static void main(String[] args) {
24
25
26
27
28
29
30

```

## Step 3: Implement multi-level inheritance

- 3.1 Let us also explore something known as **multi-level inheritance**. In this, there is a class called CC which is the child of CB, CA is the parent, CB is the child and CC is the grandchild. How it can work for us. Let's take one example over here. There is this class called phone. From the phone, you get something known as an Android phone. From the Android phone, you can also come up with a Samsung Android phone.



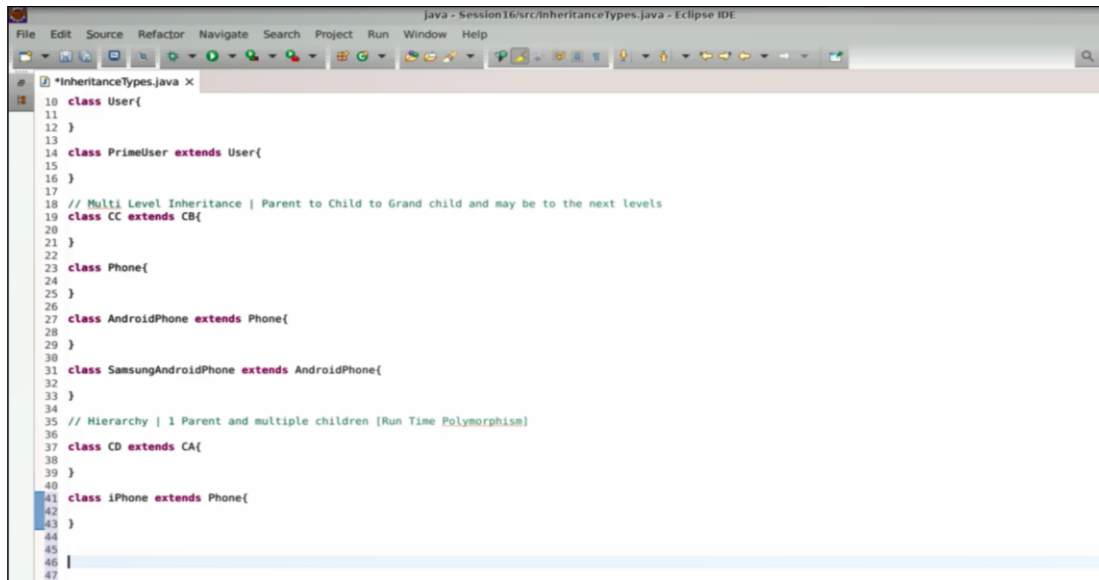
```

1 // Single Level Inheritance | 1 Parent and 1 Child
2 class CA{
3
4 }
5
6 class CB extends CA{
7
8 }
9
10 class User{
11
12 }
13
14 class PrimeUser extends User{
15
16 }
17
18 // Multi Level Inheritance | Parent to Child to Grand child and may be to the next levels
19 class CC extends CB{
20
21 }
22
23 class Phone{
24
25 }
26
27 class AndroidPhone extends Phone{
28
29 }
30
31 class SamsungAndroidPhone extends AndroidPhone{
32
33 }
34

```

## Step 4: Understand the concept of hierarchy with example data

- 4.1 Let us next see the **hierarchy**. It means one parent and multiple children. One parent and multiple children mean that there may be a class called CD, which is also the child of CA. Here now the class CA has two children CD and the CD. CB and CD become siblings to each other. From the class phone, the way you got Android phone, a class called iPhone extends the phone

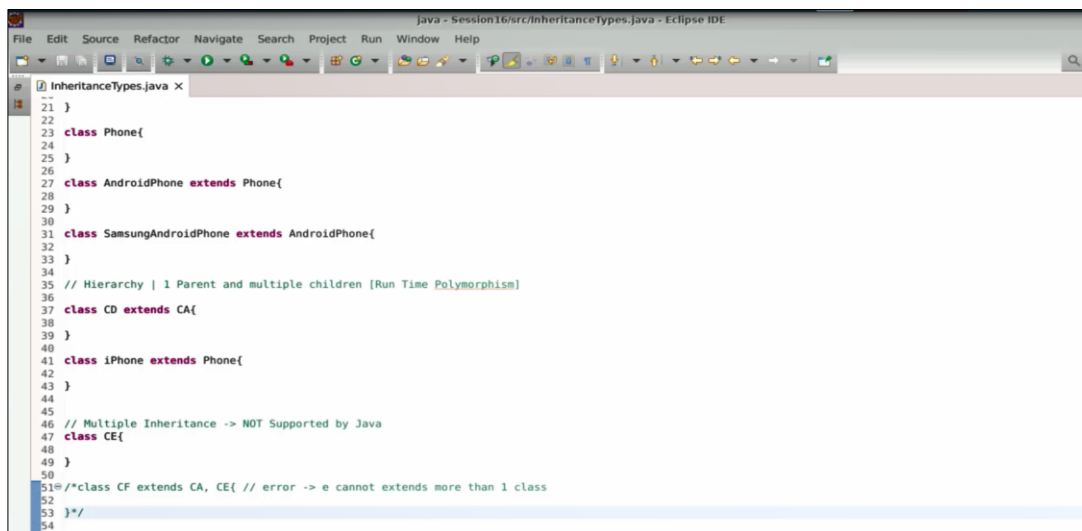


```
Java - Session16/src/inheritanceTypes.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

*inheritanceTypes.java x
10 class User{
11 }
12 }
13
14 class PrimeUser extends User{
15 }
16 }
17
18 // Multi Level Inheritance | Parent to Child to Grand child and may be to the next levels
19 class CC extends CB{
20 }
21 }
22
23 class Phone{
24 }
25 }
26
27 class AndroidPhone extends Phone{
28 }
29 }
30
31 class SamsungAndroidPhone extends AndroidPhone{
32 }
33 }
34
35 // Hierarchy | 1 Parent and multiple children [Run Time Polymorphism]
36
37 class CD extends CA{
38 }
39 }
40
41 class iPhone extends Phone{
42 }
43 }
44
45
46
47
```

## Step 5: Implement multiple inheritances with example data

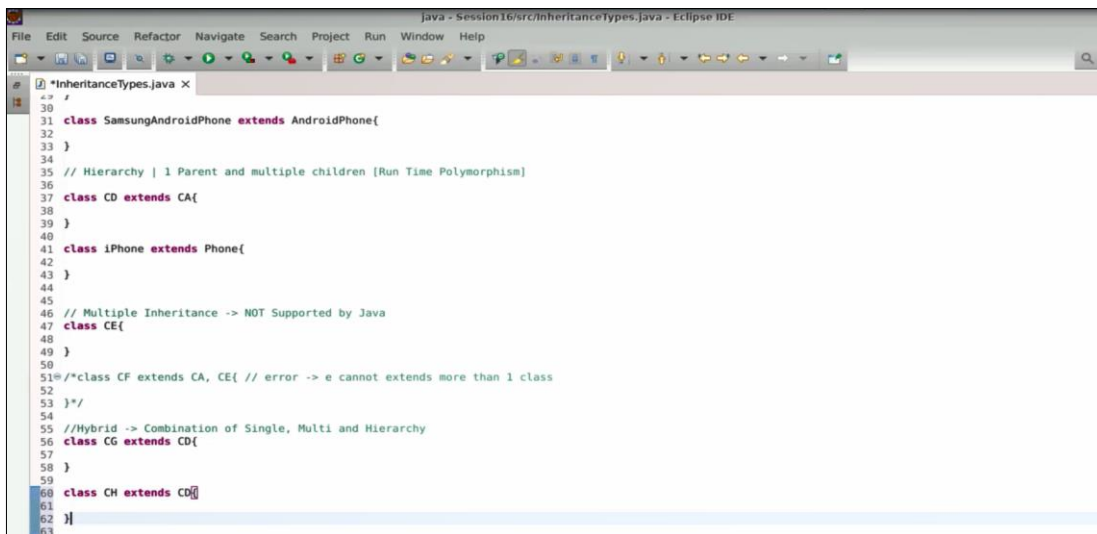
5.1 Now, the next level of inheritance is known as **multiple inheritances**. This is certainly not supported by Java. In multiple inheritances, there is more than one parent for the inheritance. Let us say there is a class called, take this class maybe by the name of, the CD you have the CE, And You are going to add a class called CF, which is going to be extending CA comma CE both, this is not allowed, this is erroneous. you cannot extend more than one class; this is something that is not supported

A screenshot of the Eclipse IDE interface. The title bar shows 'Java - Session16/src/inheritancetypes.java - Eclipse IDE'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, running, and debugging. The editor window shows the following Java code:

```
21 }
22
23 class Phone{
24 }
25 }
26
27 class AndroidPhone extends Phone{
28 }
29 }
30
31 class SamsungAndroidPhone extends AndroidPhone{
32 }
33 }
34
35 // Hierarchy | 1 Parent and multiple children [Run Time Polymorphism]
36
37 class CD extends CA{
38 }
39 }
40
41 class iPhone extends Phone{
42 }
43 }
44
45
46 // Multiple Inheritance -> NOT Supported by Java
47 class CE{
48 }
49
50
51 /*class CF extends CA, CE{ // error -> e cannot extends more than 1 class
52 }*/
53
54
```

## Step 6: Understand and use the concept of hybrid

6.1 Now, moving ahead with the **hybrid**. It is a combination of all the above, it can be a combination of Single-level, Multi-level, and hierarchy, it's sudden. It certainly goes something like this, there may be another class called CG, which extends the CD. In the same row, that's like class CH is also extending CD

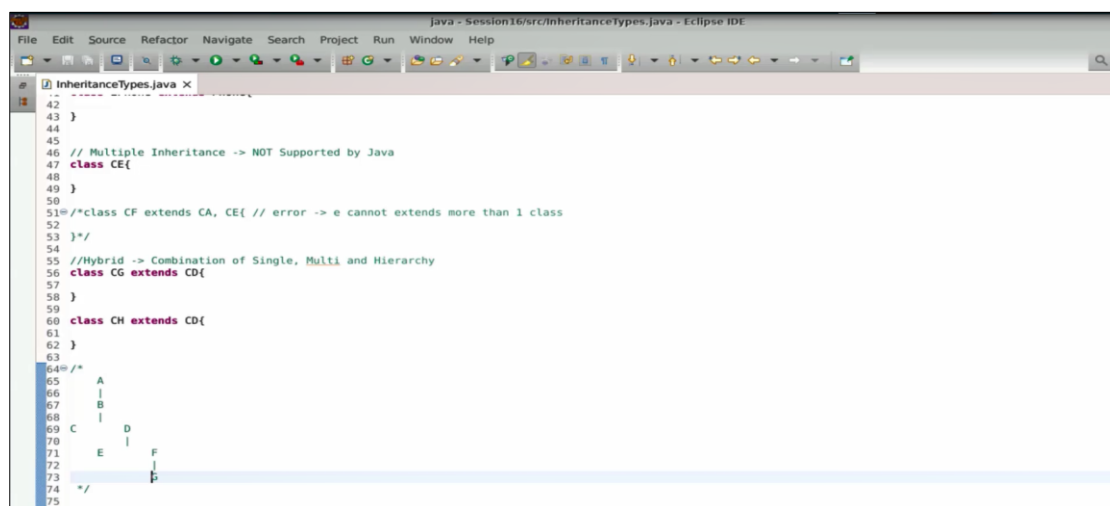


```

1  *InheritanceTypes.java x
2  //
3  //
4  //
5  //
6  //
7  //
8  //
9  //
10 //
11 //
12 //
13 //
14 //
15 //
16 //
17 //
18 //
19 //
20 //
21 //
22 //
23 //
24 //
25 //
26 //
27 //
28 //
29 //
30 //
31 class SamsungAndroidPhone extends AndroidPhone{
32 }
33 }
34 // Hierarchy | 1 Parent and multiple children [Run Time Polymorphism]
35 //
36 class CD extends CA{
37 }
38 }
39 }
40 class iPhone extends Phone{
41 }
42 }
43 }
44 //
45 //
46 // Multiple Inheritance -> NOT Supported by Java
47 class CE{
48 }
49 }
50 //
51 //class CF extends CA, CE{ // error -> e cannot extends more than 1 class
52 }
53 }
54 //
55 //Hybrid -> Combination of Single, Multi and Hierarchy
56 class CG extends CD{
57 }
58 }
59 //
60 class CH extends CD{
61 }
62 }
63 }

```

6.2 How is it going to come up, it can diagrammatically tell this part. for the hybrid you got something like parent A, then you got parent B. From B you can have C and D. Then from D you can further have something like E and maybe the F from F you can go with something like G, it's a combination, and it's all the techniques



```

42 //
43 }
44 //
45 //
46 // Multiple Inheritance -> NOT Supported by Java
47 class CE{
48 }
49 }
50 //
51 //class CF extends CA, CE{ // error -> e cannot extends more than 1 class
52 }
53 }
54 //
55 //Hybrid -> Combination of Single, Multi and Hierarchy
56 class CG extends CD{
57 }
58 }
59 //
60 class CH extends CD{
61 }
62 }
63 }
64 //
65 //
66 //
67 //
68 //
69 //
70 //
71 //
72 //
73 //
74 //
75 //

```

By following the above steps, you have successfully implemented the various types of inheritance in Java.