

Lesson 03 Demo 03

Using Inheritance in Java

Objective: The Necessity and Significance of Using Inheritance in Java

Tools: Eclipse IDE

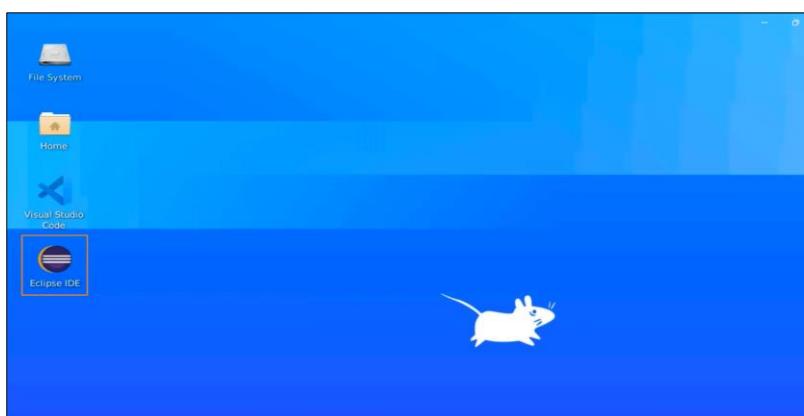
Prerequisites: None

Steps to be followed:

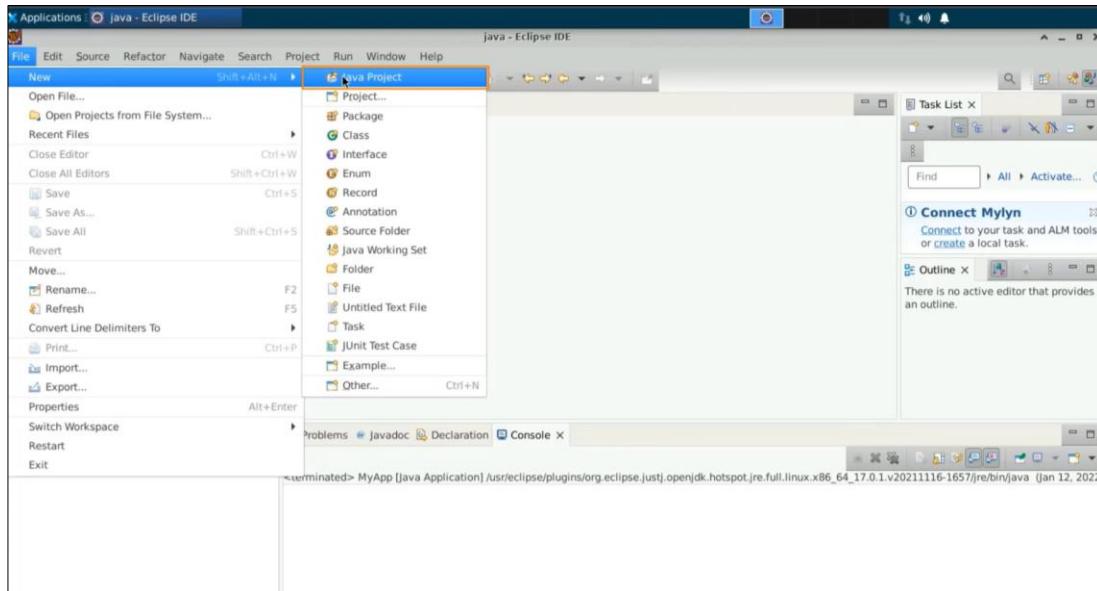
1. Open Eclipse IDE, and open a new Java project and a class
2. Create a class with three attributes
3. Create an object using the new operator and execute the code
4. Create an object with the default constructor
5. Implement the concept of sub-class and superclass
6. Access the attributes of classes
7. Execute the code with example data
8. Understand inheritance rules, their importance, and execute sample code
9. Create a default constructor and parameterized constructor
10. Initialize the arrays

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

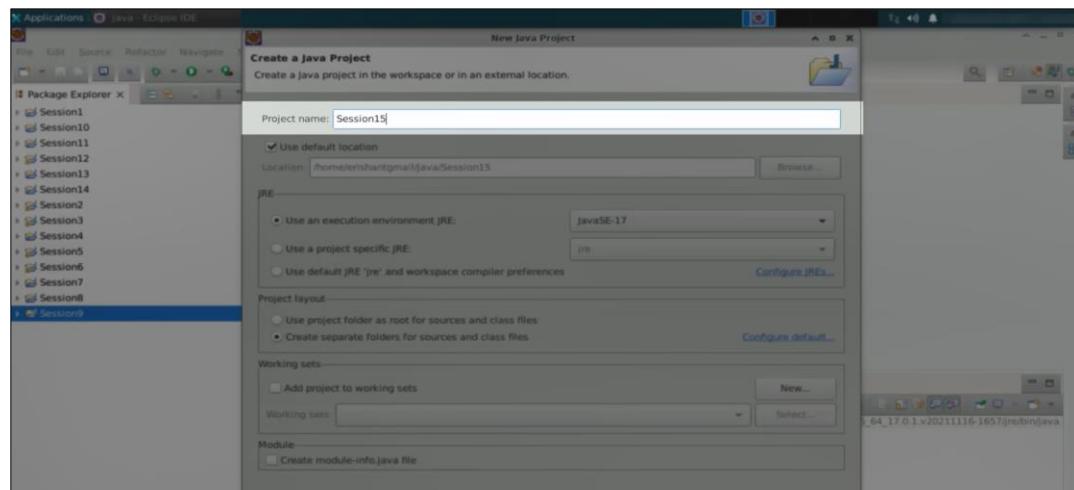
1.1 Open the Eclipse IDE



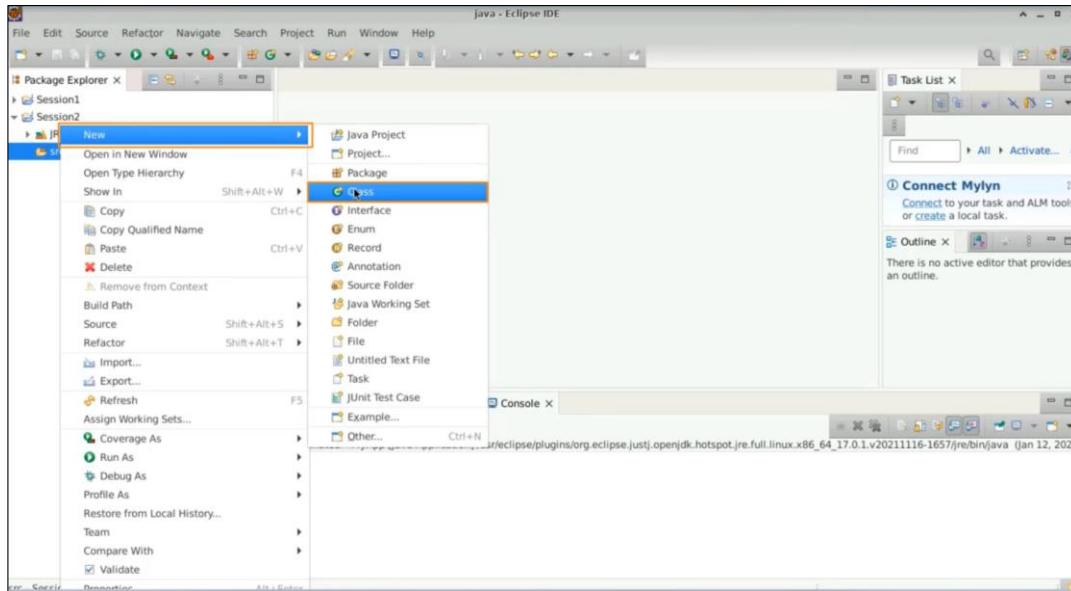
1.2. Select File, then New, and then Java project



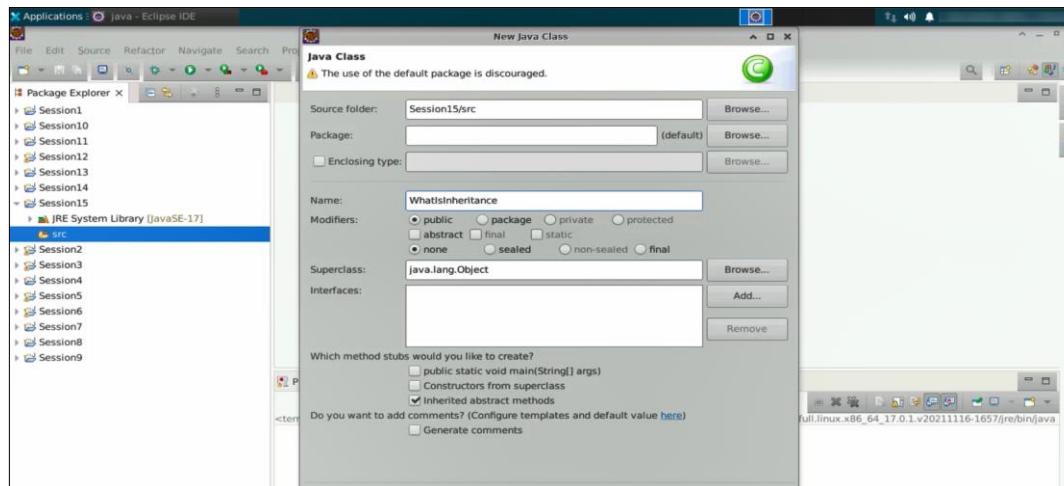
1.3 Name the project “Session15”, uncheck “Create a module info dot Java file”, and press Finish



1.4 With a Session15 on the src, do a right-click and create a new class



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



Step 2: Create a class with three attributes

2.1 Let us write a class called parent and define three attributes F name, last name, and an integer wealth. Now in the constructor of the parent initialize the first name to John, the last name to Watson, and the wealth to some 100,000. Also, use the SQL statement saying the parent, object is constructed with the default constructor

```

Java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
src WhatIsInheritance.java
1 class Parent{
2
3     String fname;
4     String lname;
5     int wealth;
6
7     Parent(){
8         fname = "John";
9         lname = "Watson";
10        wealth = 100000;
11        System.out.println("[Parent] - Object Constructed with Default Constructor");
12    }
13
14
15    public class WhatIsInheritance {
16
17        public static void main(String[] args) {
18
19
20
21
22        }
23
24    }
25

```

2.2 Now type "**show.**" The parent or simply create a method called "**showDetails.**" Here you are going to execute the print statement on the parent data. Use this like a parent, and say "**details are**" plus the name, space, the last name, and a space called wealth. This is how you create a parent class.

```

Java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
src WhatIsInheritance.java
1 class Parent{
2
3     String fname;
4     String lname;
5     int wealth;
6
7     Parent(){
8         fname = "John";
9         lname = "Watson";
10        wealth = 100000;
11        System.out.println("[Parent] - Object Constructed with Default Constructor");
12    }
13
14    void showDetails() {
15        System.out.println("[Parent] Details are: "+fname+" "+lname+" "+wealth);
16    }
17
18
19    public class WhatIsInheritance {
20
21        public static void main(String[] args) {
22
23
24            I
25
26        }
27
28    }
29

```

Step 3: Create an object using the new operator and execute the code

3.1 Now in the main method You are going to create the object of parent. Here you are with the new operator will create the object of **parent** by using the default constructor execution. Let us say you want to see the details inside the pane and object, you can simply say show the details. You will execute this method called show details

```

1  class Parent{
2
3     String fname;
4     String lname;
5     int wealth;
6
7     Parent(){
8         fname = "John";
9         lname = "Watson";
10        wealth = 100000;
11        System.out.println("[Parent] - Object Constructed with Default Constructor");
12    }
13
14    void showDetails() {
15        System.out.println("[Parent] Details are: "+fname+" "+lname+" "+wealth);
16    }
17 }
18
19
20 public class WhatIsInheritance {
21
22    public static void main(String[] args) {
23
24        Parent parent = new Parent();
25        parent.showDetails();
26    }
27 }
28
29 }
30

```

3.2 Run this code and you will see from the parent you are able to understand that there are some default construction, which is being used and it happened for the parent object. The next thing is details which is like John Watson and 100,000

```

1  class Parent{
2
3     String fname;
4     String lname;
5     int wealth;
6
7     Parent(){
8         fname = "John";
9         lname = "Watson";
10        wealth = 100000;
11        System.out.println("[Parent] - Object Constructed with Default Constructor");
12    }
13
14    void showDetails() {
15        System.out.println("[Parent] Details are: "+fname+" "+lname+" "+wealth);
16    }
17 }
18
19
20 public class WhatIsInheritance {
21
22    public static void main(String[] args) {
23
24        Parent parent = new Parent();
25        parent.showDetails();
26    }
27 }
28
29 }
30

```

<terminated> WhatIsInheritance [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/compiler.jar
[Parent] - Object Constructed with Default Constructor
[Parent] Details are: John Watson 100000

Step 4: Create an object with the default constructor

4.1 Now write one more class here and this class goes like a child. The child over here as of now, has no attributes and you are going to come up and create the constructor with a print statement, and this print statement goes like child object constructed with default constructor

```

Java - Session15/src/WhatIsInheritance.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

# WhatIsInheritance.java x
1 class Parent{
2
3     String fname;
4     String lname;
5     int wealth;
6
7     Parent(){
8         fname = "John";
9         lname = "Watson";
10        wealth = 100000;
11        System.out.println("[Parent] - Object Constructed with Default Constructor");
12    }
13
14    void showDetails() {
15        System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
16    }
17 }
18
19 class Child {
20
21     Child(){
22         System.out.println("[Child] - Object Constructed with Default Constructor");
23     }
24 }
25
26
27
28 public class WhatIsInheritance {
29
30     public static void main(String[] args) {
31
32         Parent parent = new Parent();
33         parent.showDetails();
34
35     }
36 }
37
38

```

4.2 When you create another class child, it has no attribute. There is just a constructor, so comment on the two statements in line number 32 and 33. Which means now you are not creating a parent object. You're going to create an object of a child

```

Applications Java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

# WhatIsInheritance.java x
1 class Parent{
2
3     String fname;
4     String lname;
5     int wealth;
6
7     Parent(){
8         fname = "John";
9         lname = "Watson";
10        wealth = 100000;
11        System.out.println("[Parent] - Object Constructed with Default Constructor");
12    }
13
14    void showDetails() {
15        System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
16    }
17 }
18
19 class Child {
20
21     Child(){
22         System.out.println("[Child] - Object Constructed with Default Constructor");
23     }
24 }
25
26
27
28 public class WhatIsInheritance {
29
30     public static void main(String[] args) {
31
32         //Parent parent = new Parent();
33         //parent.showDetails();
34
35         Child child = new Child();
36
37     }
38

```

4.3 Run this code and you see that there is this child object constructed with the default constructor. It is the object which has no data within it, it is an empty object. It is just a storage container with the constructor

```

Java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
# WhatIsInheritance.java x
1 class Parent{
2
3     String fname;
4     String lname;
5     int wealth;
6
7     Parent(){
8         fname = "John";
9         lname = "Watson";
10        wealth = 100000;
11        System.out.println("[Parent] - Object Constructed with Default Constructor");
12    }
13
14    void showDetails() {
15        System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
16    }
17 }
18
19 class Child {
20
21     Child(){
22         System.out.println("[Child] - Object Constructed with Default Constructor");
23     }
24 }
25
26
27
28 public class WhatIsInheritance {
29
30     public static void main(String[] args) {
31
32         //Parent parent = new Parent();
33         //parent.showDetails();
34     }
}

```

Step 5: Implement the concept of sub-class and superclass

5.1 Now if you need to relate the class parent with the class child through inheritance. You will simply come here and add that the child extends parent. You are relating to objects or two classes. Where **child** is known as the subclass, and **parent** is the superclass or the other way round

```

Java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
# WhatIsInheritance.java x
1 class Parent{
2
3     String fname;
4     String lname;
5     int wealth;
6
7     Parent(){
8         fname = "John";
9         lname = "Watson";
10        wealth = 100000;
11        System.out.println("[Parent] - Object Constructed with Default Constructor");
12    }
13
14    void showDetails() {
15        System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
16    }
17 }
18
19 // Inheritance : Create a Relationship of Parent Child using extends keyword between 2 classes/objects
20 class Child extends Parent{
21
22     Child(){
23         System.out.println("[Child] - Object Constructed with Default Constructor");
24     }
25 }
26
27
28 public class WhatIsInheritance {
29
30     public static void main(String[] args) {
31
32         //Parent parent = new Parent();
33         //parent.showDetails();
34     }
}

```

5.2 Run the code and What you see is a magical output. This is also the very **first thumb rule** to inheritance. Inheritance is a parent-child relationship. Which has a rule that before the object of the child object of the parent is constructed in memory. That is why you will get two objects constructed instead of 1 object

```

java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Refactor Navigate Search Project Run Window Help
Run WhatIsInheritance
Parent
    fname;
    lname;
    wealth;
    ...
    System.out.println("[Parent] - Object Constructed with Default Constructor");
    ...
    void showDetails() {
        System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
    }
}
Inheritance : Create a Relationship of Parent Child using extends keyword between 2 classes/objects
Child extends Parent
    ...
    System.out.println("[Child] - Object Constructed with Default Constructor");

class WhatIsInheritance {
    public static void main(String[] args) {
        //Parent parent = new Parent();
        //parent.showDetails();
    }
}

```

<terminated> WhatIsInheritance[Java Application]/usr/eclipse/plugins/org.eclipse.justj.op/[Parent] - Object Constructed with Default Constructor
[Child] - Object Constructed with Default Constructor

5.3 Rule 2 says as a child it can access attributes slash methods of the parent. In case the **child** is not going to have, some attribute or method it will look up into the **parent**, and if the parent has its child can access it

```

java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*WhatIsInheritance.java X
1 package WhatIsInheritance;
2 
3 class Parent {
4     String fname = "John";
5     String lname = "Watson";
6     int wealth = 100000;
7     ...
8     System.out.println("[Parent] - Object Constructed with Default Constructor");
9 }
10 
11 void showDetails() {
12     System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
13 }
14 }
15 
16 // Inheritance : Create a Relationship of Parent Child using extends keyword between 2 classes/objects
17 class Child extends Parent {
18     ...
19     Child() {
20         System.out.println("[Child] - Object Constructed with Default Constructor");
21     }
22 }
23 
24 /*
25 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory :)
26 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have it
27 */
28 
29 public class WhatIsInheritance {
30     public static void main(String[] args) {
31         //Parent parent = new Parent();
32         //parent.showDetails();
33         ...
34     }
35 }
36 
37 /*
38 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory :)
39 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have it
40 */
41 
42

```

5.4 Now, type "**child.showDetails.**" You need to see that you can access the **showDetails** method of the parent. You can access it if you execute it, and you will get to see the details coming up

```

File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
java - Session15/src/WhatIsInheritance.java - Eclipse IDE
Problems Javadoc Declaration Console X
<terminated> WhatIsInheritance [Java Application] /usr/eclipse/plugins/org.eclipses.jdt.core_3.11.0.v20150611-1415.jar
[Parent] - Object Constructed with Default Constructor
[Child] - Object Constructed with Default Constructor
[Parent] - Details are: John Watson 100000
I

WhatIsInheritance.java
1 package com.simplilearn;
2
3 public class Parent {
4     String fname = "John";
5     String lname = "Watson";
6     int wealth = 100000;
7     void showDetails() {
8         System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
9     }
10 }
11
12 // Inheritance : Create a Relationship of Parent Child using extends keyword between 2 classes
13 class Child extends Parent{
14     Child(){
15         System.out.println("[Child] - Object Constructed with Default Constructor");
16     }
17 }
18
19 // Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
20 // Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
21
22 /*
23 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
24 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
25 */
26
27
28 /*
29 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
30 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
31 */
32
33
34 public class WhatIsInheritance {
35
36     public static void main(String[] args) {
37
38         //Parent parent = new Parent();
39         //parent.showDetails();
40
41         Child child = new Child();
42         child.showDetails();
43
44     }
45 }

```

Step 6: Access the attributes of classes

6.1 Now, let us say the child will access the attribute of the parent called **wealth**. You are going to say "**child.wealth = parent.wealth - 5000**". You are updating the value of the wealth by subtracting 5000 from it, but this is not the child's wealth. Whenever you run the program, you will observe that the wealth has been reduced by 5000 by the child, and this change is reflected in the parent

```

File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
java - Session15/src/WhatIsInheritance.java - Eclipse IDE
Problems Javadoc Declaration Console X
<terminated> WhatIsInheritance [Java Application] /usr/eclipse/plugins/org.eclipses.jdt.core_3.11.0.v20150611-1415.jar
[Parent] - Object Constructed with Default Constructor
[Child] - Object Constructed with Default Constructor
[Parent] - Details are: John Watson $5000
I

WhatIsInheritance.java
1 package com.simplilearn;
2
3 public class Parent {
4     String fname = "John";
5     String lname = "Watson";
6     int wealth = 100000;
7     void showDetails() {
8         System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
9     }
10 }
11
12 // Inheritance : Create a Relationship of Parent Child using extends keyword between 2 classes
13 class Child extends Parent{
14     Child(){
15         System.out.println("[Child] - Object Constructed with Default Constructor");
16     }
17 }
18
19 // Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
20 // Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
21
22 /*
23 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
24 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
25 */
26
27
28 /*
29 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
30 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
31 */
32
33
34 public class WhatIsInheritance {
35
36     public static void main(String[] args) {
37
38         //Parent parent = new Parent();
39         //parent.showDetails();
40
41         Child child = new Child();
42         child.wealth -= 5000;
43         child.showDetails();
44
45     }
46 }

```

6.2 Now, what if the child has those attributes? Let us add that the child has an attribute called first name, an attribute called wealth, and an additional attribute called company name

```

java - Session15/src/WhatIsInheritance.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

* WhatIsInheritance.java X
  1 package WhatIsInheritance;
  2
  3 public class Parent {
  4     String lname;
  5     int wealth;
  6     String companyName;
  7
  8     Parent() {
  9         lname = "Watson";
 10         wealth = 100000;
 11         System.out.println("[Parent] - Object Constructed with Default Constructor");
 12     }
 13
 14     void showDetails() {
 15         System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
 16     }
 17 }
 18
 19 // Inheritance : Create a Relationship of Parent Child using extends keyword between 2 classes/objects
 20 class Child extends Parent{
 21
 22     String fname;
 23     int wealth;
 24     String companyName;
 25
 26     Child(){
 27         System.out.println("[Child] - Object Constructed with Default Constructor");
 28     }
 29
 30 }
 31
 32/*
 33 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory :
 34 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have it
 35 *
 36 */
 37
 38 public class WhatIsInheritance {
 39
 40     public static void main(String[] args) {
 41         //Parent parent = new Parent();
 42
 43     }
 44 }

```

6.3 In the constructor, add the **fname** as "**Fionna**". You will work on the **lname** (last name), which is "**Watson**". Set the wealth to 50,000 and the **companyName** to "**ABC Ventures**"

```

java - Session15/src/WhatIsInheritance.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

* WhatIsInheritance.java X
  1 package WhatIsInheritance;
  2
  3 public class Parent {
  4     String lname;
  5     int wealth;
  6     String companyName;
  7
  8     Parent() {
  9         lname = "Watson";
 10         wealth = 100000;
 11         System.out.println("[Parent] - Object Constructed with Default Constructor");
 12     }
 13
 14     void showDetails() {
 15         System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
 16     }
 17 }
 18
 19 // Inheritance : Create a Relationship of Parent Child using extends keyword between 2 classes/objects
 20 class Child extends Parent{
 21
 22     String fname;
 23     int wealth;
 24     String companyName;
 25
 26     Child(){
 27         fname = "Fionna";
 28         wealth = 50000;
 29         companyName = "ABC Ventures";
 30         System.out.println("[Child] - Object Constructed with Default Constructor");
 31     }
 32
 33 }
 34
 35/*
 36 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory :
 37 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have it
 38 *
 39 */
 40
 41 public class WhatIsInheritance {
 42
 43     }
 44 }

```

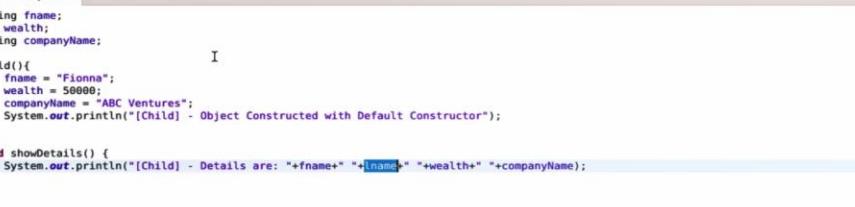
6.4 When you execute the show details method in the parent you will see that the parents' wealth remains unimpacted. When the child has its own attribute or method child going to access its own

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** java - Session15/src/WhatIsInheritance.java - Eclipse IDE
- File Menu:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Margin:** Shows line numbers from 18 to 51.
- Code Editor:** Displays Java code for creating inheritance relationships between Parent and Child classes. The code includes constructor logic and a main method.
- Output View:** Shows the execution results of the code:
 - [Parent] - Object Constructed with Default Constructor
 - [Child] - Object Constructed with Default Constructor
 - [Parent] - Details are: John Watson 100000
- Bottom Status Bar:** Informs about the current state: Terminated > WhatIsInheritance [Java Application] /usr/eclipse/plugins/org

6.5 Let us type **void showDetails()** {

```
System.out.println("[Child] - Details are: " + fname + " " + lname + " " + wealth + " " +  
companyName);}
```



The screenshot shows the Eclipse IDE interface with the title bar "java - Session15/src/WhatIsInheritance.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc. The code editor displays the following Java code:

```
File Edit Source Refactor Navigate Search Project Run Window Help

# WhatIsInheritance.java X
1
22 String name;
23 int wealth;
24 String companyName;
25
26 Child(){
27     name = "Fionna";
28     wealth = 50000;
29     companyName = "ABC Ventures";
30     System.out.println("[Child] - Object Constructed with Default Constructor");
31 }
32
33 void showDetails() {
34     System.out.println("[Child] - Details are: "+fname+" "+name+" "+wealth+" "+companyName);
35 }
36
37 }
38
39 */
40 Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory :)
41 Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
42
43 */
44
45 public class WhatIsInheritance {
46
47     public static void main(String[] args) {
48
49         //Parent parent = new Parent();
50         //parent.showDetails();
51
52         Child child = new Child();
53         child.wealth -= 5000;
54         child.showDetails();
55     }
56 }
```

Step 7: Execute the code with example data

7.1 Run the code and you can see details are Fiona Watson with 45,000 and ABC Ventures.

This beautiful concept is also referred to as **method overriding**

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** java - Session15/src/WhatIsInheritance.java - Eclipse IDE
- Left Panel (Outline View):** Shows the class structure with Child and WhatIsInheritance classes.
- Code Editor:**

```

1 String fname;
2 int wealth;
3 String companyName;
4
5 Child(){
6     fname = "Fiona";
7     wealth = 50000;
8     companyName = "ABC Ventures";
9     System.out.println("[Child] - Object Constructed with Default Constructor");
10 }
11
12 void showDetails() {
13     System.out.println("[Child] - Details are: "+fname+" "+lname+" "+wealth+" "+companyName);
14 }
15
16 /**
17 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
18 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
19 */
20
21 public class WhatIsInheritance {
22     public static void main(String[] args) {
23         //Parent parent = new Parent();
24         //parent.showDetails();
25         Child child = new Child();
26         child.wealth -= 5000;
27         child.showDetails();
28     }
29 }
```
- Console View:** Displays the output of the executed code:


```
<terminated> WhatIsInheritance [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core_3.11.0.v20150611-1410.jar
[Parent] - Object Constructed with Default Constructor
[Child] - Object Constructed with Default Constructor
[Child] - Details are: Fiona Watson 45000 ABC Ventures
```

7.2 You can use this reference variable called super which refers to the parent object and You can execute the method called show details. Now this is the method of the parent. You are now exiting the method of the parents

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** java - Session15/src/WhatIsInheritance.java - Eclipse IDE
- Left Panel (Outline View):** Shows the class structure with Child and WhatIsInheritance classes.
- Code Editor:**

```

1 String fname;
2 int wealth;
3 String companyName;
4
5 Child(){
6     fname = "Fiona";
7     wealth = 50000;
8     companyName = "ABC Ventures";
9     System.out.println("[Child] - Object Constructed with Default Constructor");
10 }
11
12 // Method Overriding
13 void showDetails() {
14     super.showDetails(); // method of the Parent
15     System.out.println("[Child] - Details are: "+fname+" "+lname+" "+wealth+" "+companyName);
16 }
17
18 /**
19 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory :)
20 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
21 */
22
23 public class WhatIsInheritance {
24     public static void main(String[] args) {
25         //Parent parent = new Parent();
26         //parent.showDetails();
27         Child child = new Child();
28         child.wealth -= 5000;
29     }
30 }
```

7.3 Run the code, what you will notice is the parent days would come up as John Watson and no impact on the wealth, but child details coming up with the Watson being inherited from the parent and the wealth has been impacted.

```

java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
# WhatIsInheritance.java X
22     String fname;
23     int wealth;
24     String companyName;
25
26     Child(){}
27     fname = "Fionna";
28     wealth = 50000;
29     companyName = "ABC Ventures";
30     System.out.println("[Child] - Object Constructed with Default Constructor");
31 }
32
33 // Method Overriding
34 void showDetails() {
35     super.showDetails(); // method of the Parent
36     System.out.println("[Child] - Details are: "+fname+" "+lname+" "+wealth+" "+companyName);
37 }
38
39 /*
40 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
41 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
42 */
43
44 public class WhatIsInheritance {
45
46     public static void main(String[] args) {
47         //Parent parent = new Parent();
48         //parent.showDetails();
49
50         Child child = new Child();
51         child.wealth -= 5000;
52     }
53 }

```

<terminated> WhatIsInheritance [Java Application] /usr/eclipse/plugins/org.eclipsesource.jdt.core/compiler/1.1.0/jdtCompilerAdapter
[Parent] - Object Constructed with Default Constructor
[Child] - Object Constructed with Default Constructor
[Parent] - Details are: John Watson 100000
[Child] - Details are: Fionna Watson 45000 ABC Ventures

7.4 If the child is not going to have an attribute called wealth, when you run the code, what you would see is that the wealth is going to be inherited from the parent. What belongs to the child, would be accessed. If something is not even there in the parent, then definitely you will get a compile-time error

```

java - Session15/src/WhatIsInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
# WhatIsInheritance.java X
22     String fname;
23     //int wealth;
24     String companyName;
25
26     Child(){}
27     fname = "Fionna";
28     //wealth = 50000;
29     companyName = "ABC Ventures";
30     System.out.println("[Child] - Object Constructed with Default Constructor");
31 }
32
33 // Method Overriding
34 void showDetails() {
35     super.showDetails(); // method of the Parent
36     System.out.println("[Child] - Details are: "+fname+" "+lname+" "+wealth+" "+companyName);
37 }
38
39 /*
40 * Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory
41 * Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
42 */
43
44 public class WhatIsInheritance {
45
46     public static void main(String[] args) {
47         //Parent parent = new Parent();
48         //parent.showDetails();
49
50         Child child = new Child();
51         child.wealth -= 5000;
52     }
53 }

```

<terminated> WhatIsInheritance [Java Application] /usr/eclipse/plugins/org.eclipsesource.jdt.core/compiler/1.1.0/jdtCompilerAdapter
[Parent] - Object Constructed with Default Constructor
[Child] - Object Constructed with Default Constructor
[Parent] - Details are: John Watson 95000
[Child] - Details are: Fionna Watson 95000 ABC Ventures

Step 8: Understand inheritance rules, their importance, and execute sample code

8.1 One last rule of inheritance, which is typical with the link of rule 2. If a parent has any property marked as private. It will not be accessible to the child

```

java - Session15/src/WhatIsInheritance.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

public class WhatIsInheritance {
    void showDetails() {
        System.out.println("[Parent] - Details are: "+fname+" "+lname+" "+wealth);
    }
}

// Inheritance : Create a Relationship of Parent Child using extends keyword between 2 classes/objects
class Child extends Parent{
    String fname;
    int wealth;
    String companyName;
    Child(){
        fname = "Fionna";
        wealth = 50000;
        companyName = "ABC Ventures";
        System.out.println("[Child] - Object Constructed with Default Constructor");
    }
}

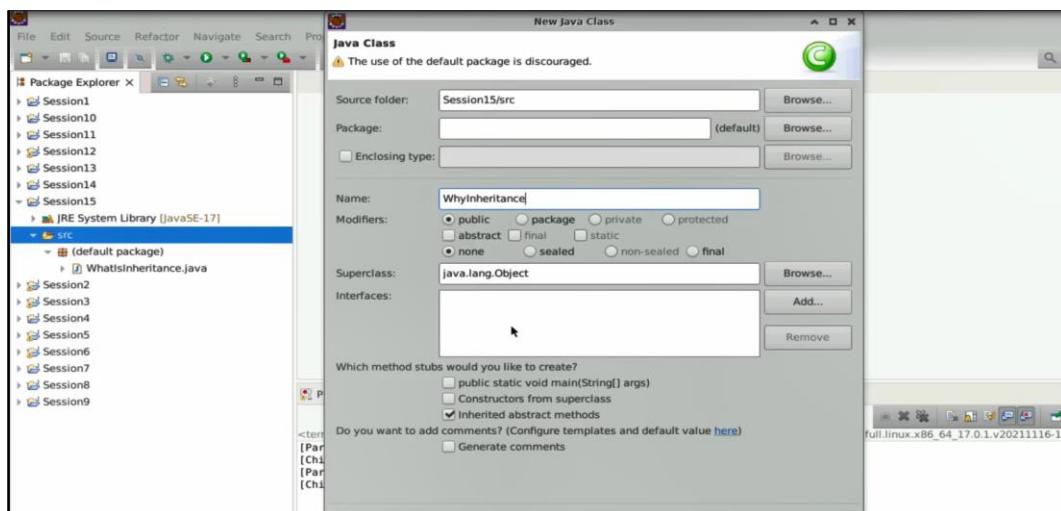
// Method Overriding
void showDetails() {
    super.showDetails(); // method of the Parent
    System.out.println("[Child] - Details are: "+fname+" "+lname+" "+wealth+" "+companyName);
}

}

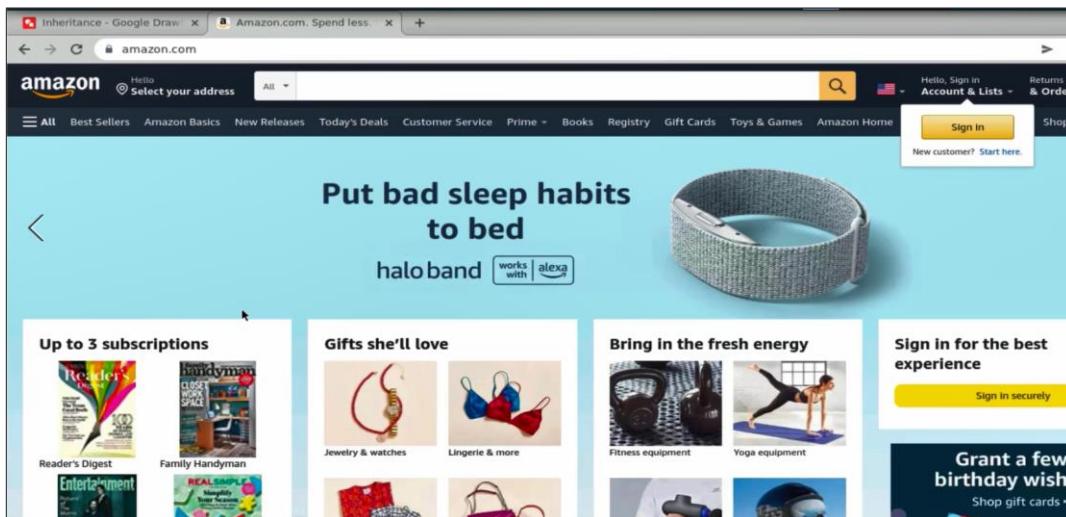
/*
Inheritance Rule1: Before the Object of Child, Object of Parent is constructed in memory :)
Inheritance Rule2: As a Child, it can access attributes/methods of the Parent if child does not have its own
Inheritance Rule3: If parent has any property marked as private, it will not be accessible by the child
*/
public class WhatIsInheritance {

```

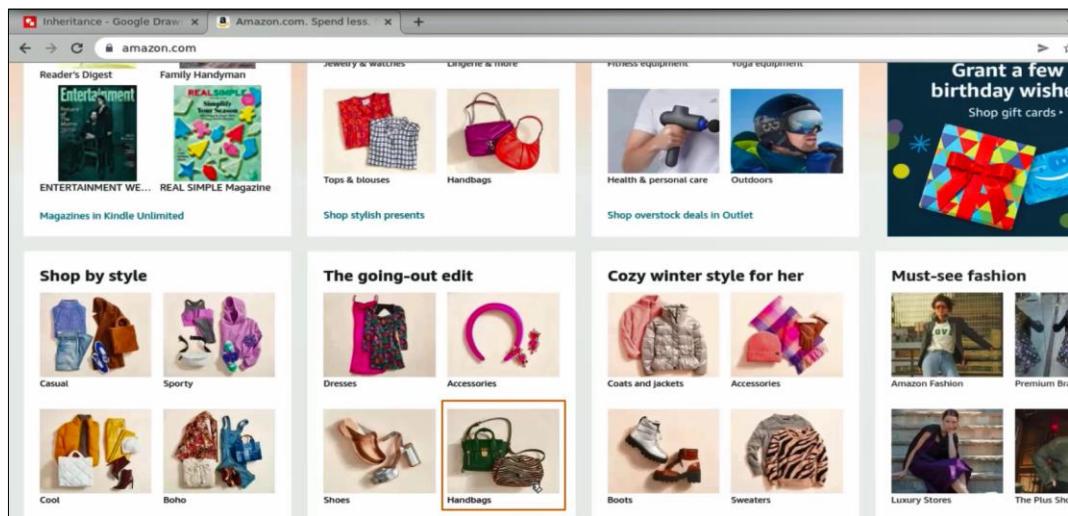
8.2 Now let us proceed ahead and see how the inheritance has a real significance. You are going to write a new class Known as **WhyInheritance** With the main method



8.3 Let us take a use case of an e-commerce store. You are going to open something known as Amazon



8.4 Now on Amazon you do have various products to be worked on. As you can see there is a product called dress, accessory, shoe, handbag. How is this all managed.



8.5 You will write a class and first You will represent something known as a shoe. To represent a shoe, you need to have an id for the issue. The name and description of the issue. Then there is another attribute called price. Now You need to have something on as an integer array of sizes, brand, and colors

```

File Edit Source Refactor Navigate Search Project Run Window Help
java - Session15/src/WhyInheritance.java - Eclipse IDE
WhyInheritance.java X
1  class Shoe{
2      int id;
3      String name;
4      String description;
5      int price;
6      String brand;
7      int[] sizes;
8      String[] colors;
9  }
10 }
11
12 public class WhyInheritance {
13
14     public static void main(String[] args) {
15
16
17
18
19
20
21
}

```

8.6 Now write a class called television it would have some common attributes id, name, description, integer price, screen size, array of sizes, array of colors, and ledtechnology

```

File Edit Source Refactor Navigate Search Project Run Window Help
java - Session15/src/WhyInheritance.java - Eclipse IDE
WhyInheritance.java X
1  class Shoe{
2      int id;
3      String name;
4      String description;
5      int price;
6      String brand;
7      int[] sizes;
8      String[] colors;
9  }
10 }
11
12 class TV{
13     int id;
14     String name;
15     String description;
16     int price;
17     int screenSize;
18     String color;
19     String ledTechnology;
20 }
21
22 public class WhyInheritance {
23
24     public static void main(String[] args) {
25
26
27
28
29
30
31
}

```

8.7 Let us consider one more object on the E-commerce platform. This is a mobile device or a mobile phone. For a phone, it will have an id, name, description, price, brand, ram, storage, and operating system

```

File Edit Source Refactor Navigate Search Project Run Window Help
*WhyInheritance.java X
1 class Shoe{
2     int id;
3     String name;
4     String description;
5     int price;
6     String brand;
7     int[] sizes;
8     String[] colors;
9 }
10 }
11 class TV{
12     int id;
13     String name;
14     String description;
15     int price;
16     String brand;
17     int screenSize;
18     String color;
19     String ledTechnology;
20 }
21 }
22 class MobilePhone{
23     int id;
24     String name;
25     String description;
26     int price;
27     String brand;
28     int ram;
29     int storage;
30     String os;
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }

```

8.8 Now, if you are noticing a pattern, the first four or five attributes they are common across all the different objects. when you know that there is similarity when you must create different objects that is where you need inheritance. Create a class known as product. make these common attributes to be a part of product. Eliminate the common attributes. Say shoe extends product, television extends the product, and mobile phone extends the product

```

File Edit Source Refactor Navigate Search Project Run Window Help
*WhyInheritance.java X
1 /*
2 */
3 class Product{
4     int id;
5     String name;
6     String description;
7     int price;
8     String brand;
9 }
10 }
11 class Shoe extends Product{
12     int[] sizes;
13     String[] colors;
14 }
15 }
16 class TV extends Product{
17     int screenSize;
18     String color;
19     String ledTechnology;
20 }
21 }
22 class MobilePhone extends Product{
23     int ram;
24     int storage;
25     String os;
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 public class WhyInheritance {
70 }

```

8.9 For the product here, you can use a default constructor or a parameterized constructor.

You can add void show the product. You are going to display id, name, add print description and then you can come up with the price and let us say brand. These same attribute initializations and these attributes display part

```

java - Session15/src/WhyInheritance.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

* WhyInheritance.java X
  41 String name;
  42 String description;
  43 int price;
  44 String brand;
  45
  46 Product(){
  47 }
  48
  49
  50 public Product(int id, String name, String description, int price, String brand) {
  51     this.id = id;
  52     this.name = name;
  53     this.description = description;
  54     this.price = price;
  55     this.brand = brand;
  56 }
  57
  58 void showProduct() {
  59     System.out.println("-----Product Details-----");
  60     System.out.println(id+" "+name);
  61     System.out.println(description);
  62     System.out.println(price+" "+brand);
  63     System.out.println("-----");
  64 }
  65
  66
  67 class Shoe extends Product{
  68
  69     int[] sizes;
  70     String[] colors;
  71 }
  72
  73 class TV extends Product{
  74
  75     int screenSize;
  76     String color;
  77     String ledTechnology;
  78 }
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  
```

Step 9: Create a default constructor and parameterized constructor

9.1 Let us show you how so you can create a shoe default constructor. And for my parameterized constructor, you will just say source, generate the constructor from the superclass. You will type generate and what you see is a magical part, for the shoe you will take all the attributes. But for the attributes called ID, name, description, Price, and the brand, this would be, right away passed to the parent for the initialization

```

java - Session15/src/WhyInheritance.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

* * WhyInheritance.java X
  55     this.brand = brand;
  56
  57
  58 void showProduct() {
  59     System.out.println("-----Product Details-----");
  60     System.out.println(id+" "+name);
  61     System.out.println(description);
  62     System.out.println(price+" "+brand);
  63     System.out.println("-----");
  64 }
  65
  66
  67 class Shoe extends Product{
  68
  69     int[] sizes;
  70     String[] colors;
  71
  72     Shoe(){
  73
  74 }
  75
  76     public Shoe(int id, String name, String description, int price, String brand, int[] sizes, String[] colors) {
  77         super(id, name, description, price, brand);
  78     }
  79
  80
  81
  82
  83 class TV extends Product{
  84
  85     int screenSize;
  86     String color;
  87     String ledTechnology;
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  
```

Step 10: Initialize the arrays

10.1 Regarding my array of sizes or my array of colors, you can initialize them like this: **this.sizes** and **this.colors**. A point to notice now is that the attribute initialization has reduced drastically. In previous examples, you had to initialize all these attributes, but now you are just initializing 2 of them

```

Java - Session15/src/WhyInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
e WhyInheritance.java X
56     1
57
58     void showProduct() {
59         System.out.println("-----Product Details-----");
60         System.out.println(id+" "+name);
61         System.out.println(description);
62         System.out.println(price+" "+brand);
63         System.out.println("-----");
64     }
65 }
66
67 class Shoe extends Product{
68
69     int[] sizes;
70     String[] colors;
71
72     Shoe(){
73
74     }
75
76     public Shoe(int id, String name, String description, int price, String brand, int[] sizes, String[] colors) {
77         super(id, name, description, price, brand);
78         this.sizes = sizes;
79         this.colors = colors;
80     }
81
82
83 }
84
85 class TV extends Product{
86
87     int screenSize;
88     String color;
89     String ledTechnology;
90
91     TV(){
92
93     }
94
95     public TV(int id, String name, String description, int price, String brand, int screenSize, String color, String ledTechnology) {
96         super(id, name, description, price, brand);
97         this.screenSize = screenSize;
98         this.color = color;
99         this.ledTechnology = ledTechnology;
100    }
101
102
103
104 }
105
106 class MobilePhone extends Product{
107
108     int ram;
109     int storage;
110     String os;

```

10.2 Let us take it from the television. Create a constructor through the parent class. Execute the parent constructor, and then set **this.screenSize** to screenSize. Then you have **this.color** as color. After that, set **this.ledTechnology** to ledTechnology. That is how you save development time

```

Java - Session15/src/WhyInheritance.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
e WhyInheritance.java X
56     1
57
58     void showProduct() {
59         System.out.println("-----Product Details-----");
60         System.out.println(id+" "+name);
61         System.out.println(description);
62         System.out.println(price+" "+brand);
63         System.out.println("-----");
64     }
65 }
66
67 class Shoe extends Product{
68
69     int[] sizes;
70     String[] colors;
71
72     Shoe(){
73
74     }
75
76     public Shoe(int id, String name, String description, int price, String brand, int[] sizes, String[] colors) {
77         super(id, name, description, price, brand);
78         this.sizes = sizes;
79         this.colors = colors;
80     }
81
82
83 }
84
85 class TV extends Product{
86
87     int screenSize;
88     String color;
89     String ledTechnology;
90
91     TV(){
92
93     }
94
95     public TV(int id, String name, String description, int price, String brand, int screenSize, String color, String ledTechnology) {
96         super(id, name, description, price, brand);
97         this.screenSize = screenSize;
98         this.color = color;
99         this.ledTechnology = ledTechnology;
100    }
101
102
103
104 }
105
106 class MobilePhone extends Product{
107
108     int ram;
109     int storage;
110     String os;

```

10.3 For the mobile phone, you can now right-click and select **Source**, then **Generate Constructor from Parent**. Take your additional attributes and put them inside the constructor. Here you go: **this.ram = ram**, **this.storage = storage**, and **this.os = os**

```

java - Session15/src/WhyInheritance.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

WhyInheritance.java X
89     String ledTechnology;
90
91     TV(){
92
93 }
94
95     public TV(int id, String name, String description, int price, String brand, int screenSize, String color, String ledTechnology) {
96         super(id, name, description, price, brand);
97         this.screenSize = screenSize;
98         this.color = color;
99         this.ledTechnology = ledTechnology;
100
101
102 }
103
104     class MobilePhone extends Product{
105
106         int ram;
107         int storage;
108         String os;
109
110         MobilePhone(){
111
112 }
113
114
115     public MobilePhone(int id, String name, String description, int price, String brand, int ram, int storage, String os) {
116         super(id, name, description, price, brand);
117         this.ram = ram;
118         this.storage = storage;
119         this.os = os;
120     }
121
122

```

10.4 Now again, create a method called "**void showShoe.**" Inside the method, you would display the additional attributes, such as sizes and colors. You can simply call "**showProduct**" in the same way for the television and mobile phone. You now understand the use case of inheritance, the rules it comes with, and why you need it. You need inheritance because it saves development time. In addition to this, inheritance provides several benefits, including the ability to implement design patterns with the help of a concept called runtime polymorphism

```

java - Session15/src/WhyInheritance.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

WhyInheritance.java X
53     this.description = description;
54     this.price = price;
55     this.brand = brand;
56 }
57
58     void showProduct() {
59         System.out.println("-----Product Details-----");
60         System.out.println(id+" "+name);
61         System.out.println(description);
62         System.out.println(price+" "+brand);
63         System.out.println("-----");
64     }
65
66
67     class Shoe extends Product{
68
69         int[] sizes;
70         String[] colors;
71
72     Shoe(){
73
74 }
75
76     public Shoe(int id, String name, String description, int price, String brand, int[] sizes, String[] colors) {
77         super(id, name, description, price, brand);
78         this.sizes = sizes;
79         this.colors = colors;
80     }
81
82     void showShoe() {
83         showProduct();
84         System.out.println(sizes);
85         System.out.println(colors);
86     }
87

```

By following the above steps, you have successfully used Inheritance in Java and understood its necessity and significance.