Rajan Gautam

19BCP101

Div. II, CE 19

SOT, PDPU

Pandit Deendayal Energy University

School of Technology

Design Pattern and Thinking (20CP210P)

B. Tech - Computer Science & Engineering (Sem-IV)

Table of Contents

Lab 1 Assignment: Factory Design Pattern for Creation of OS for Phones

AIM: **To write a Java Program to implement Factory Design Pattern for Creation of OS for Phones**

CODE:

MainFactory.java

```
1. package Phone_OS;
2.
3. import java.util.Scanner;
4.
5. public class MainFactory {
6.
7.   public static void main(String[] args) {
8.           OS_Factory myOS = new OS_Factory();
9.           Scanner scanner = new Scanner(System.in);
10.
11.                  System.out.println("<--- Select by Developer --->");
12.                  System.out.println("1. Apple");
13.                  System.out.println("2. Google");
14.                  System.out.println("3. Microsoft");
15.                  System.out.println("4. KaiOS");
16.                  System.out.println("5. Samsung");
17.                  System.out.print("Enter your choice: ");
18.
19.                  String choice = scanner.nextLine();
20.                  scanner.close();
21.
22.                  OS obj = myOS.getInstance(choice);
23.
24.                  obj.name();
25.                  obj.specs();
26.                  obj.secure();
27.                  obj.developer();
28.                  obj.messaging();
29.                  obj.source();
30.                  obj.written_language();
31.                  obj.kernel_type();
32.                  obj.working_state();
33.
34.          }
35.
36.  }
```

OS_Factory.java

```
1. package Phone_OS;
2.
3. public class OS_Factory {
4.   public OS getInstance(String str)
5.     {
6.           if (str.equals("Google")) {
7.                 return new Android();
8.           }
9.
10.          if (str.equals("Apple")) {
11.                return new IOS();
12.          }
13.
14.          if (str.equals("KaiOS")) {
15.                return new KaiOS();
16.          }
17.
18.          if (str.equals("Microsoft")) {
19.                return new Windows();
20.          }
21.
22.          if (str.equals("Samsung")) {
23.                return new Tizen();
24.          }
25.
26.          return null;
27.      }
28.  }
```

OS.java

```
1. package Phone_OS;
2.
3. public interface OS
4. {
5.          public void name();
6.          public void specs();
7.          public void secure();
8.          public void source();
9.          public void developer();
10.         public void messaging();
11.         public void written_language();
12.         public void kernel_type();
13.         public void working_state();
14.  }
```

Android.java

```
1. package Phone_OS;
2.
3. public class Android implements OS
4. {
5.          @Override
6.          public void name() {
7.                  System.out.println("<--Android-->");
8.          }
9.
10.                 @Override
11.         public void specs() {
12.                 System.out.println("Most used phone OS.");
13.         }
14.
15.         @Override
16.         public void secure() {
17.                 System.out.println("Moderately Secured.");
18.
19.         }
20.
21.         @Override
22.         public void source() {
23.                 System.out.println("Source Model: Open Source.");
24.
25.         }
26.
27.         @Override
28.         public void developer() {
29.                 System.out.println("Product of Google.");
30.
31.         }
32.
33.         @Override
34.         public void messaging() {
35.                 System.out.println("Provides Message app for messaging.");
36.
37.         }
38.
39.         @Override
40.         public void written_language() {
41.                 System.out.println("Written Language: Java, C, C++.");
42.
43.         }
44.
45.         @Override
46.         public void kernel_type() {
47.                 System.out.println("Kernel Type: Linux");
48.
49.         }
50.
51.         @Override
```

```
52.         public void working_state() {
53.                 System.out.println("Working State: Current");
54.
55.         }
56.  }
```

IOS.java

```
1. package Phone_OS;
2.
3. public class IOS implements OS
4. {
5.         @Override
6.         public void name() {
7.         System.out.println("<--IOS-->");
8.         }
9.
10.        @Override
11.        public void specs()
12.        {
13.                System.out.println("Most secure phone OS.");
14.        }
15.
16.        @Override
17.        public void secure() {
18.                System.out.println("Most Secured Phone.");
19.
20.        }
21.        @Override
22.        public void source() {
23.                System.out.println("Source Model: Closed Source.");
24.
25.        }
26.
27.        @Override
28.        public void developer() {
29.                System.out.println("Product of Apple.");
30.
31.        }
32.
33.        @Override
34.        public void messaging() {
35.                System.out.println("Provides iMessage for messaging.");
36.
37.        }
38.
39.        @Override
40.        public void written_language() {
41.                System.out.println("Written Language: C, C++, Swift.");
42.
43.        }
```

```
44.
45.         @Override
46.         public void kernel_type() {
47.                 System.out.println("Kernel Type: Hybrid");
48.
49.         }
50.
51.         @Override
52.         public void working_state() {
53.                 System.out.println("Working State: Current");
54.
55.         }
56.  }
```

Windows.java

```
1. package Phone_OS;
2.
3. public class Windows implements OS
4. {
5.         @Override
6.         public void name() {
7.         System.out.println("<--Windows-->");
8.         }
9.
10.         @Override
11.         public void specs() {
12.                 System.out.println("I am about to die.");
13.         }
14.
15.         @Override
16.         public void secure() {
17.                 System.out.println("Not so Secured.");
18.
19.         }
20.         @Override
21.         public void source() {
22.                 System.out.println("Source Model: Closed Source.");
23.
24.         }
25.
26.         @Override
27.         public void developer() {
28.                 System.out.println("Product of Microsoft.");
29.
30.         }
31.
32.         @Override
33.         public void messaging() {
34.                 System.out.println("User needs to download separate app
   for messaging.");
```

```
35.
36.         }
37.
38.         @Override
39.         public void written_language() {
40.                 System.out.println("Written Language: C, C++.");
41.
42.         }
43.
44.         @Override
45.         public void kernel_type() {
46.                 System.out.println("Kernel Type: Hybrid.");
47.
48.         }
49.
50.         @Override
51.         public void working_state() {
52.                 System.out.println("Working State: Discontinued");
53.
54.         }
55.  }
```

KaiOS.java

```
1. package Phone_OS;
2.
3. public class KaiOS implements OS
4. {
5.          @Override
6.          public void name() {
7.          System.out.println("<--KaiOS-->");
8.          }
9.
10.         @Override
11.         public void specs()
12.         {
13.                 System.out.println("Emerging OS.");
14.         }
15.
16.         @Override
17.         public void secure() {
18.                 System.out.println("Moderately Secured OS.");
19.
20.         }
21.         @Override
22.         public void source() {
23.                 System.out.println("Source Model: Open Source.");
24.
25.         }
26.
27.         @Override
```

```
28.          public void developer() {
29.                  System.out.println("Product of KaiOS Technologies.");
30.
31.          }
32.
33.          @Override
34.          public void messaging() {
35.                  System.out.println("Provide inbuilt messaging app.");
36.
37.          }
38.
39.          @Override
40.          public void written_language() {
41.                  System.out.println("Written Language: HTML, CSS,
   JavaScript.");
42.
43.          }
44.
45.          @Override
46.          public void kernel_type() {
47.                  System.out.println("Kernel Type: Monolithic");
48.
49.          }
50.
51.          @Override
52.          public void working_state() {
53.                  System.out.println("Working State: Current");
54.
55.          }
56.  }
```

Tizen.java

```
1. package Phone_OS;
2.
3. public class Tizen implements OS
4. {
5.          @Override
6.          public void name() {
7.          System.out.println("<--Tizen-->");
8.          }
9.
10.         @Override
11.         public void specs() {
12.                 System.out.println("Electronic Devices' OS.");
13.
14.         }
15.
16.         @Override
17.         public void secure() {
18.                 System.out.println("Moderately Secured.");
```

```
19.
20.         }
21.
22.         @Override
23.         public void source() {
24.                 System.out.println("Source Model: Open Source.");
25.
26.         }
27.
28.         @Override
29.         public void developer() {
30.                 System.out.println("Product of Samsung.");
31.
32.         }
33.
34.         @Override
35.         public void messaging() {
36.                 System.out.println("Provides Messages app for
    messaging.");
37.
38.         }
39.
40.         @Override
41.         public void written_language() {
42.                 System.out.println("Written Language: HTML5, C, C++.");
43.
44.         }
45.
46.         @Override
47.         public void kernel_type() {
48.                 System.out.println("Kernel Type: Monolithic.");
49.
50.         }
51.
52.         @Override
53.         public void working_state() {
54.                 System.out.println("Working State: Current");
55.
56.         }
57.  }
```

OUTPUT:

```
<--- Select by Developer --->
1. Apple
2. Google
3. Microsoft
4. KaiOS
5. Samsung
Enter your choice: Microsoft
<--Windows-->
I am about to die.
Not so Secured.
Product of Microsoft.
User needs to download separate app for messaging.
Source Model: Closed Source.
Written Language: C, C++.
Kernel Type: Hybrid.
Working State: Discontinued
```

```
<--- Select by Developer --->
1. Apple
2. Google
3. Microsoft
4. KaiOS
5. Samsung
Enter your choice: Apple
<--IOS-->
Most secure phone OS.
Most Secured Phone.
Product of Apple.
Provides iMessage for messaging.
Source Model: Closed Source.
Written Language: C, C++, Swift.
Kernel Type: Hybrid
Working State: Current
```