

Shadan Khan 1876267 SDP lab3

Factory Pattern

```
EXPLORER  ...  J EnemyTesting.java  J EnemyShip.java  J EnemyShipFactory.java  J RocketEnemyShip.java  J UFOEnemyShip.java  J BigUFOEnemyShip.java

OPEN EDITORS
src > J EnemyTesting.java > EnemyTesting > main(String[])
3  public class EnemyTesting {
4
5      Run | Debug
6      public static void main(String[] args) {
7
8          // Create ship
9          EnemyShipFactory shipFactory = new EnemyShipFactory();
10
11          // Enemy ship object
12          EnemyShip theEnemy = null;
13
14          Scanner userInput = new Scanner(System.in);
15
16          System.out.println("What type of ship? (Ufo , Rocket , BigUfo)");
17          if (userInput.hasNextLine()) {
18              String typeOfShip = userInput.nextLine();
19
20              theEnemy = shipFactory.getShip(typeOfShip);
21
22              if (theEnemy != null) {
23                  doStuffEnemy(theEnemy);
24              }
25              else {
26                  System.out.print("Please enter U, R, or B next time");
27              }
28          }
29          userInput.close();
30      }
31
32      // Executes methods of the super class
33
34      public static void doStuffEnemy(EnemyShip anEnemyShip) {
35
36          anEnemyShip.displayEnemyShip();
37
38          anEnemyShip.followHeroShip();
39
40          anEnemyShip.enemyShipShoots();
41      }
42  }
```

```
EXPLORER  ...  J EnemyTesting.java  J EnemyShip.java  J EnemyShipFactory.java  J RocketEnemyShip.java  J UFOEnemyShip.java  J BigUFOEnemyShip.java

OPEN EDITORS
src > J EnemyShip.java > ...
1  |
2  public abstract class EnemyShip {
3
4      private String name;
5      private double amtDamage;
6
7      public String getName() {
8          return name;
9      }
10
11      public void setName(String newName) {
12          name = newName;
13      }
14
15      public double getDamage() {
16          return amtDamage;
17      }
18
19      public void setDamage(double newDamage) {
20          amtDamage = newDamage;
21      }
22
23      public void followHeroShip() {
24          System.out.println(getName() + " is following the hero");
25      }
26
27      public void displayEnemyShip() {
28          System.out.println(getName() + " is on the screen");
29      }
30
31      public void enemyShipShoots() {
32          System.out.println(getName() + " attacks and does " + getDamage() + " damage to hero");
33      }
34
35  }
36  }
```

```
src > J EnemyShipFactory.java > %s EnemyShipFactory > getShip(String)
1 public class EnemyShipFactory {
2     public EnemyShip getShip(String shipType) {
3         if (shipType == null) {
4             return null;
5         }
6
7         if (shipType.equalsIgnoreCase(anotherString: "ROCKET")) {
8             return new RocketEnemyShip();
9         } else if (shipType.equalsIgnoreCase(anotherString: "UFO")) {
10            return new UFOEnemyShip();
11        } else if (shipType.equalsIgnoreCase(anotherString: "BIGUFO")) {
12            return new BigUFOEnemyShip();
13        }
14
15        return null;
16    }
17
18    public void followHeroShip() {
19    }
20
21    public void displayEnemyShip() {
22    }
23
24    public void enemyShipShoots() {
25    }
26
27 }
28
```

The git repository at /Users/kam

```
src > J RocketEnemyShip.java > ...
1
2 public class RocketEnemyShip extends EnemyShip {
3
4
5     public RocketEnemyShip(){
6
7         setName(newName: "Rocket Enemy Ship");
8
9         setDamage(newDamage: 10.0);
10    }
11
12 }
13
14
15
16
```

EXPLORER ... J EnemyTesting.java J EnemyShip.java J EnemyShipFactory.java J RocketEnemyShip.java J UFOEnemyShip.java X J BigUFOEnemyShip.java

OPEN EDITORS

- J EnemyTesting.j...
- J EnemyShip.jav...
- J EnemyShipFact...
- J RocketEnemyS...
- X J UFOEnemyShip...
- J BigUFOEnemy...

FACTORYLAB

- > bin
- src
 - J BigUFOEnemyShip...
 - J BigUFOEnemyShip...
 - J EnemyShip.class
 - J EnemyShip.java
 - J EnemyShipFactory...
 - J EnemyShipFactory...
 - J EnemyTesting.class
 - J EnemyTesting.java
 - J RocketEnemyShip...
 - J RocketEnemyShip.j...
 - J UFOEnemyShip.cla...
 - J UFOEnemyShip.java
- ~\$FactoryMethodLa...

src > J UFOEnemyShip.java > ...

```
1
2 public class UFOEnemyShip extends EnemyShip{
3
4     public UFOEnemyShip(){
5
6         setName(newName: "UFO Enemy Ship");
7
8         setDamage(newDamage: 20.0);
9
10    }
11
12
13
14 }
15
```

EXPLORER ... J EnemyTesting.java J EnemyShip.java J EnemyShipFactory.java J RocketEnemyShip.java J UFOEnemyShip.java J BigUFOEnemyShip.java X

OPEN EDITORS

- J EnemyTesting.j...
- J EnemyShip.jav...
- J EnemyShipFact...
- J RocketEnemyS...
- J UFOEnemyShip...
- X J BigUFOEnemy...

FACTORYLAB

- > bin
- src
 - J BigUFOEnemyShip...
 - J BigUFOEnemyShip...
 - J EnemyShip.class
 - J EnemyShip.java
 - J EnemyShipFactory...
 - J EnemyShipFactory...
 - J EnemyTesting.class
 - J EnemyTesting.java
 - J RocketEnemyShip...
 - J RocketEnemyShip.j...
 - J UFOEnemyShip.cla...
 - J UFOEnemyShip.java
- ~\$FactoryMethodLa...

src > J BigUFOEnemyShip.java > ...

```
1
2 public class BigUFOEnemyShip extends UFOEnemyShip{
3
4     public BigUFOEnemyShip(){
5
6         setName(newName: "Big UFO Enemy Ship");
7
8         setDamage(newDamage: 40.0);
9
10    }
11
12
13 }
```

Output:

```

src > J EnemyTesting.java > Run | Debug
3 public class EnemyTesting {
4
5 public static void main(String[] args) {
6
7     // Create ship
8     EnemyShipFactory shipFactory = new EnemyShipFactory();
9
10    // Enemy ship object
11    EnemyShip theEnemy = null;
12
13    Scanner userInput = new Scanner(System.in);

```

```

cd "/Users/kamal/Downloads/FactoryLab/src/" && javac EnemyTesting.java && java EnemyTesting
Shadan@Shadans-MacBook-Pro FactoryLab % cd "/Users/kamal/Downloads/FactoryLab/src/" && javac EnemyTesting.java && java EnemyTesting
What type of ship? (Ufo , Rocket , BigUfo)
Ufo
UFO Enemy Ship is on the screen
UFO Enemy Ship is following the hero
UFO Enemy Ship attacks and does 20.0 damage to hero
Shadan@Shadans-MacBook-Pro src %

```

Abstract pattern:

```

src > J Client.java > Run | Debug
1 public class Client {
2
3
4 public static void main(String[] args) {
5
6     Computer pc = FactoryProducer.createComputer(new PCFactory(ram: "2 GB", hdd: "500 GB", cpu: "2.4 GHz"));
7
8     System.out.print("\n PC Factory:\n" + pc);
9
10    Computer server = FactoryProducer.createComputer(new ServerFactory(ram: "16 GB", hdd: "1 TB", cpu: "2.9 GHz"));
11
12    System.out.print("\n \n Server Factory:\n" + server);
13
14 }
15
16
17

```

EXPLORER

OPEN EDITORS

- Client.java src
- ComputerFacto...
- FactoryProduc...
- Computer.java...
- PCFactory.java...
- PC.java src
- ServerFactory.j...
- Server.java src

ABSTRACTFACTORYLAB

- bin
- src
 - Client.class
 - Client.java
 - Computer.class
 - Computer.java
 - ComputerFactory.j...
 - FactoryProducer.java
 - PC.class
 - PC.java
 - PCFactory.class
 - PCFactory.java
 - Server.class
 - Server.java
 - ServerFactory.class
 - ServerFactory.java

src > ComputerFactory.java > ...

```
1
2 public abstract class ComputerFactory {
3
4     abstract Computer createComputer();
5
6 }
7
```

EXPLORER

OPEN EDITORS

- Client.java src
- ComputerFacto...
- FactoryProduc...
- Computer.java...
- PCFactory.java...
- PC.java src
- ServerFactory.j...
- Server.java src

ABSTRACTFACTORYLAB

- bin
- src
 - Client.class
 - Client.java
 - Computer.class
 - Computer.java
 - ComputerFactory.j...
 - FactoryProducer.java
 - PC.class
 - PC.java
 - PCFactory.class
 - PCFactory.java
 - Server.class
 - Server.java
 - ServerFactory.class
 - ServerFactory.java

src > FactoryProducer.java > ...

```
1 public class FactoryProducer {
2     public static Computer createComputer(ComputerFactory factory) {
3         return factory.createComputer();
4     }
5 }
6
```

EXPLORER ... J Client.java J ComputerFactory.java J FactoryProducer.java J Computer.java X J PCFactory.java J PC.java

OPEN EDITORS

- J Client.java src
- J ComputerFacto...
- J FactoryProduc...
- X J Computer.java...
- J PCFactory.java...
- J PC.java src
- J ServerFactory.j...
- J Server.java src

ABSTRACTFACTORYLAB

- > bin
- > src
 - J Client.class
 - J Client.java
 - J Computer.class
 - J Computer.java
 - J ComputerFactory.j...
 - J FactoryProducer.java
 - J PC.class
 - J PC.java
 - J PCFactory.class
 - J PCFactory.java
 - J Server.class
 - J Server.java
 - J ServerFactory.class
 - J ServerFactory.java

src > J Computer.java > ...

```
1
2 public abstract class Computer {
3     public abstract String getRAM();
4
5     public abstract String getHDD();
6
7     public abstract String getCPU();
8
9     @Override
10    public String toString() {
11        return "RAM=" + this.getRAM() + ", HDD=" + this.getHDD() + ", CPU=" + this.getCPU();
12    }
13 }
14
```

EXPLORER ... J Client.java J ComputerFactory.java J FactoryProducer.java J Computer.java J PCFactory.java X J PC.java

OPEN EDITORS

- J Client.java src
- J ComputerFacto...
- J FactoryProduc...
- J Computer.java...
- X J PCFactory.java...
- J PC.java src
- J ServerFactory.j...
- J Server.java src

ABSTRACTFACTORYLAB

- > bin
- > src
 - J Client.class
 - J Client.java
 - J Computer.class
 - J Computer.java
 - J ComputerFactory.j...
 - J FactoryProducer.java
 - J PC.class
 - J PC.java
 - J PCFactory.class
 - J PCFactory.java
 - J Server.class
 - J Server.java
 - J ServerFactory.class
 - J ServerFactory.java

src > J PCFactory.java > PCFactory > createComputer()

```
1
2 public class PCFactory extends ComputerFactory {
3
4     private String ram;
5     private String hdd;
6     private String cpu;
7
8     public PCFactory(String ram, String hdd, String cpu) {
9         this.ram = ram;
10        this.hdd = hdd;
11        this.cpu = cpu;
12    }
13
14    @Override
15    public Computer createComputer() {
16        return new PC(ram, hdd, cpu);
17    }
18 }
19
```

```
src > J PC.java > PC > PC(String, String, String)
1
2 public class PC extends Computer {
3
4     private String ram;
5     private String hdd;
6     private String cpu;
7
8     public PC(String ram, String hdd, String cpu) {
9         this.ram = ram;
10        this.hdd = hdd;
11        this.cpu = cpu;
12    }
13
14    @Override
15    public String getRAM() {
16        return this.ram;
17    }
18
19    @Override
20    public String getHDD() {
21        return this.hdd;
22    }
23
24    @Override
25    public String getCPU() {
26        return this.cpu;
27    }
28
29 }
30
```

```
src > J ServerFactory.java > ServerFactory
1
2 public class ServerFactory extends ComputerFactory {
3
4     private String ram;
5     private String hdd;
6     private String cpu;
7
8     public ServerFactory(String ram, String hdd, String cpu) {
9         this.ram = ram;
10        this.hdd = hdd;
11        this.cpu = cpu;
12    }
13
14    @Override
15    public Computer createComputer() {
16        return new PC(ram, hdd, cpu);
17    }
18
19 }
```



```
src > J Server.java > Server
1
2 public class Server extends Computer {
3
4     private String ram;
5     private String hdd;
6     private String cpu;
7
8     public Server(String ram, String hdd, String cpu) {
9         this.ram = ram;
10        this.hdd = hdd;
11        this.cpu = cpu;
12    }
13
14    @Override
15    public String getRAM() {
16        return this.ram;
17    }
18
19    @Override
20    public String getHDD() {
21        return this.hdd;
22    }
23
24    @Override
25    public String getCPU() {
26        return this.cpu;
27    }
28
29 }
```

Output:

```
1 public class Client {
2
3     Run | Debug
4     public static void main(String[] args) {
5
6         Computer pc = FactoryProducer.createComputer(new PCFactory("2 GB", "500 GB", "2.4 GHz"));
7
8         System.out.print("\n PC Factory:\n" + pc);
9
10        Computer server = FactoryProducer.createComputer(new ServerFactory("16 GB", "1 TB", "2.9 GHz"));
11
12        System.out.print("\n \n Server Factory:\n" + server);
13
14    }
15 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
cd "/Users/kamal/Downloads/AbstractFactoryLab/src/" && javac Client.java && java Client
Shadan@Shadans-MacBook-Pro AbstractFactoryLab % cd "/Users/kamal/Downloads/AbstractFactoryLab/src/" && javac Client.java && java Client

PC Factory:
RAM= 2 GB, HDD=500 GB, CPU=2.4 GHz

Server Factory:
RAM= 16 GB, HDD=1 TB, CPU=2.9 GHz
Shadan@Shadans-MacBook-Pro src %
```