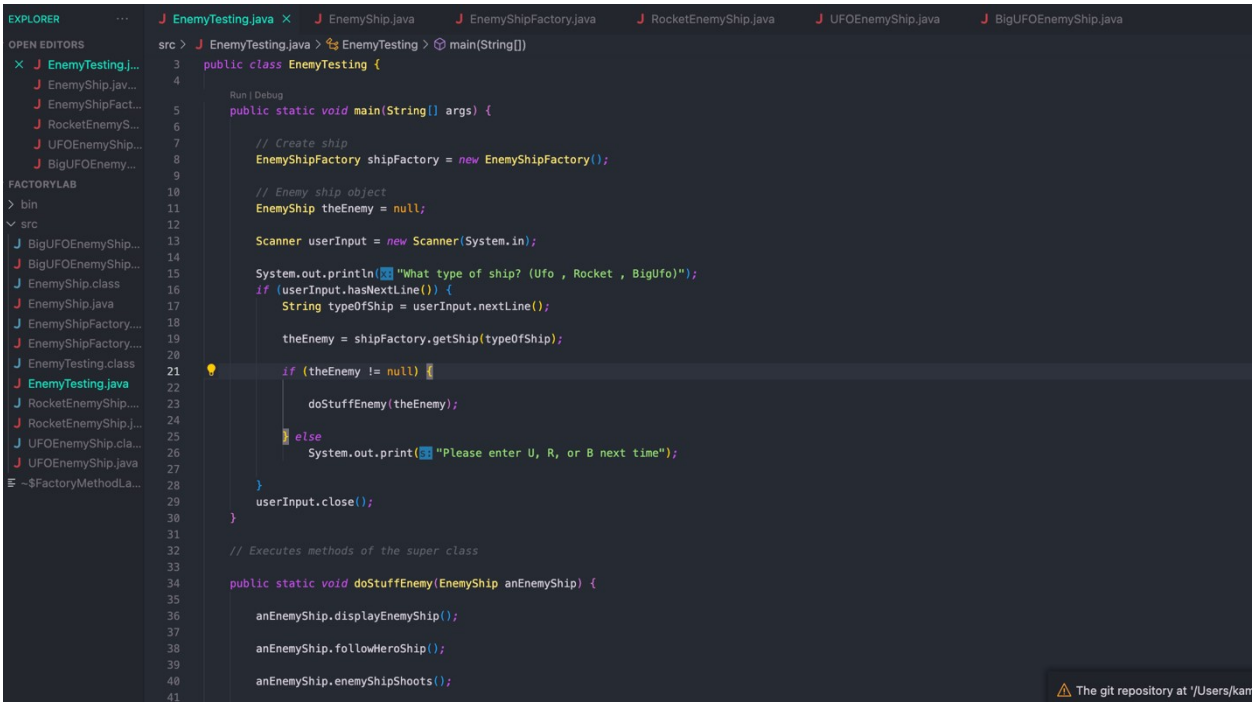
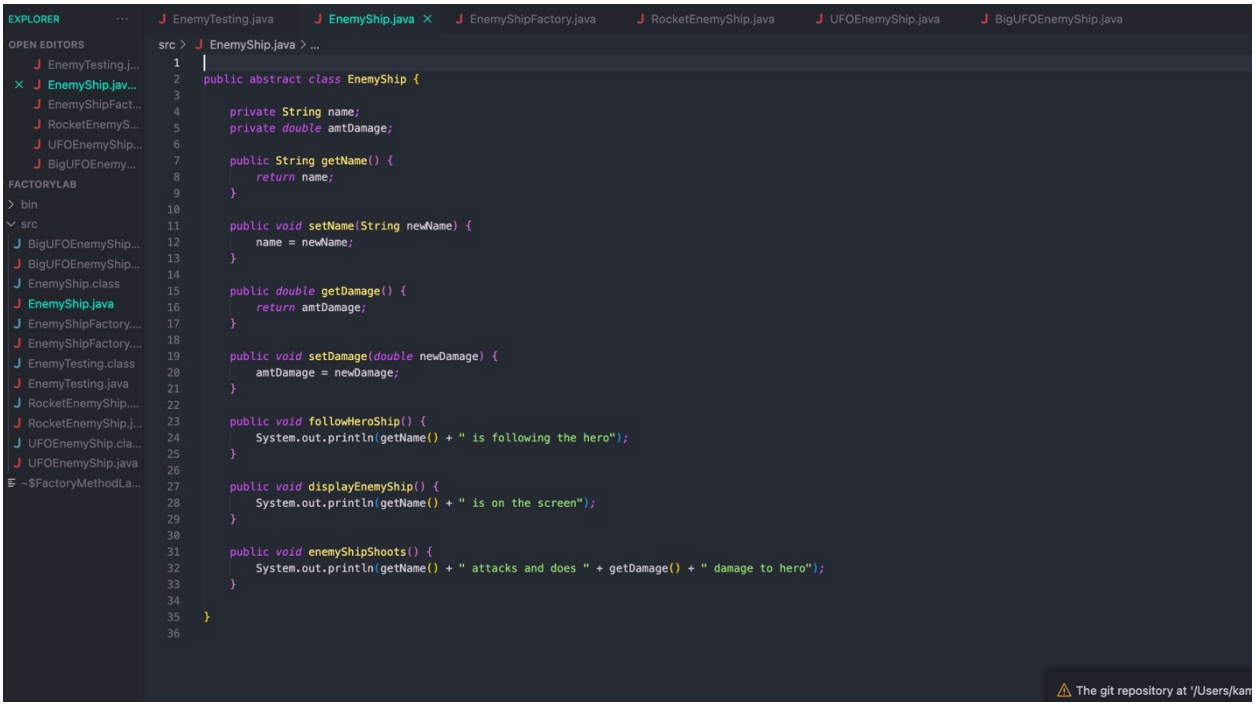
Shadan Khan 1876267 SDP lab3

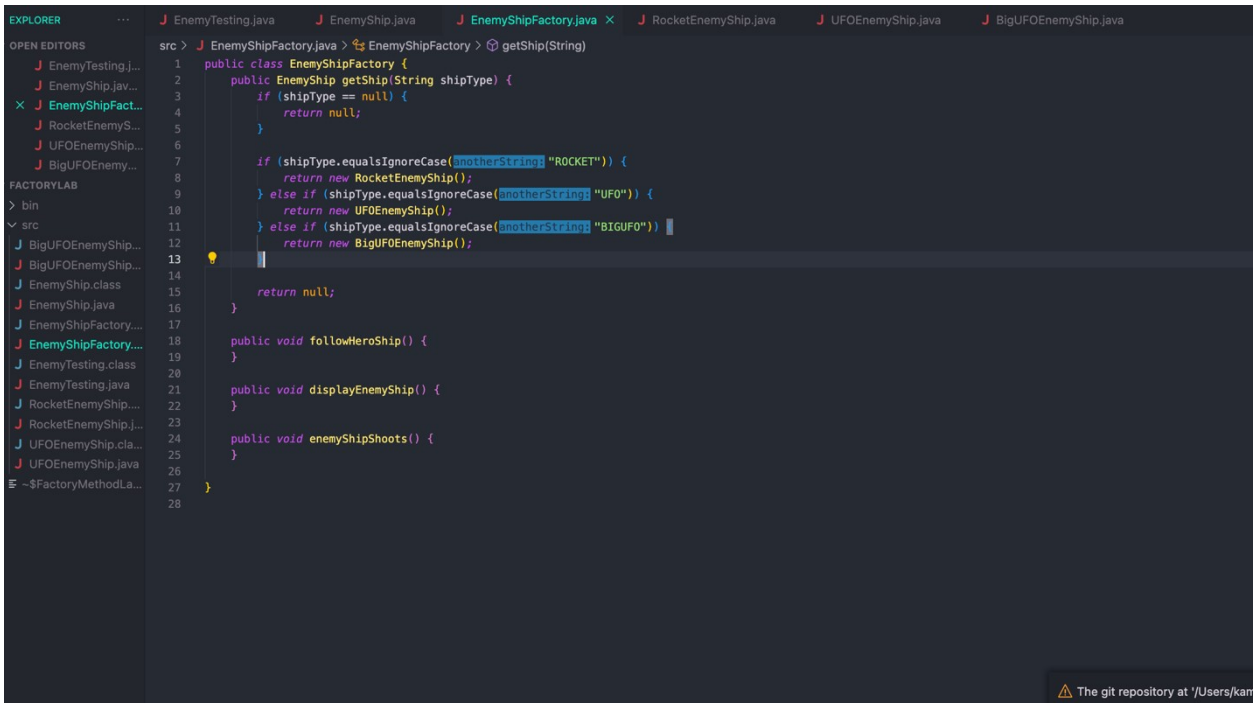
Factory Pattern

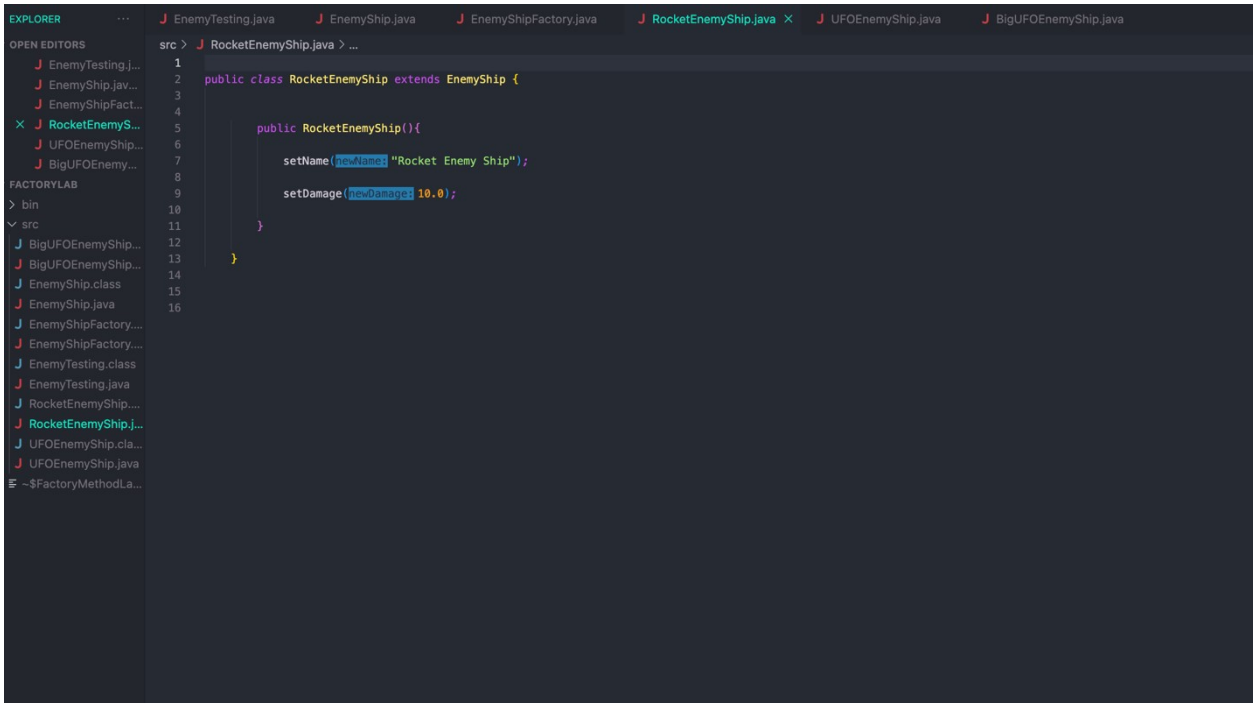


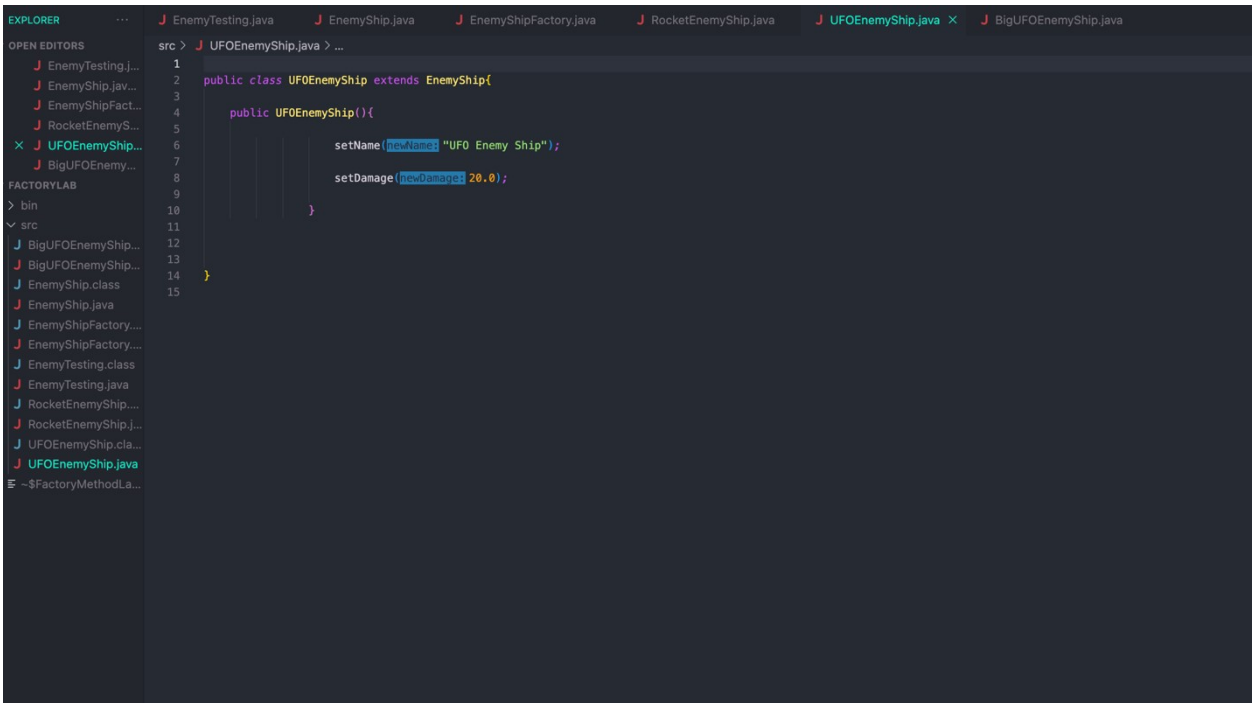
```java
public class EnemyTesting {

    Run | Debug
    public static void main(String[] args) {

        // Create ship
        EnemyShipFactory shipFactory = new EnemyShipFactory();

        // Enemy ship object
        EnemyShip theEnemy = null;

        Scanner userInput = new Scanner(System.in);

        System.out.println( x: "What type of ship? (Ufo , Rocket , BigUfo)");
        if (userInput.hasNextLine()) {
            String typeOfShip = userInput.nextLine();

            theEnemy = shipFactory.getShip(typeOfShip);

            if (theEnemy != null) {

                doStuffEnemy(theEnemy);

            } else
                System.out.print( x: "Please enter U, R, or B next time");

        }
        userInput.close();
    }

    // Executes methods of the super class

    public static void doStuffEnemy(EnemyShip anEnemyShip) {

        anEnemyShip.displayEnemyShip();

        anEnemyShip.followHeroShip();

        anEnemyShip.enemyShipShoots();
```

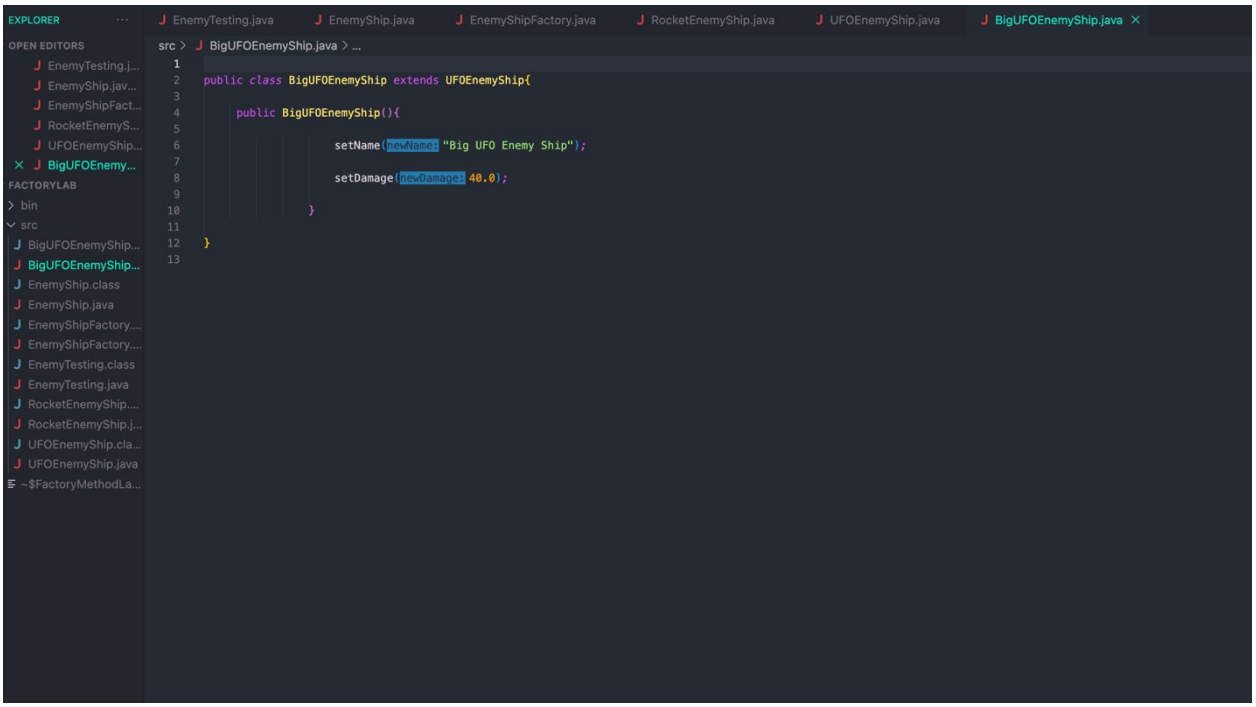


```java
public abstract class EnemyShip {

    private String name;
    private double amtDamage;

    public String getName() {
        return name;
    }

    public void setName(String newName) {
        name = newName;
    }

    public double getDamage() {
        return amtDamage;
    }

    public void setDamage(double newDamage) {
        amtDamage = newDamage;
    }

    public void followHeroShip() {
        System.out.println(getName() + " is following the hero");
    }

    public void displayEnemyShip() {
        System.out.println(getName() + " is on the screen");
    }

    public void enemyShipShoots() {
        System.out.println(getName() + " attacks and does " + getDamage() + " damage to hero");
    }

}
```

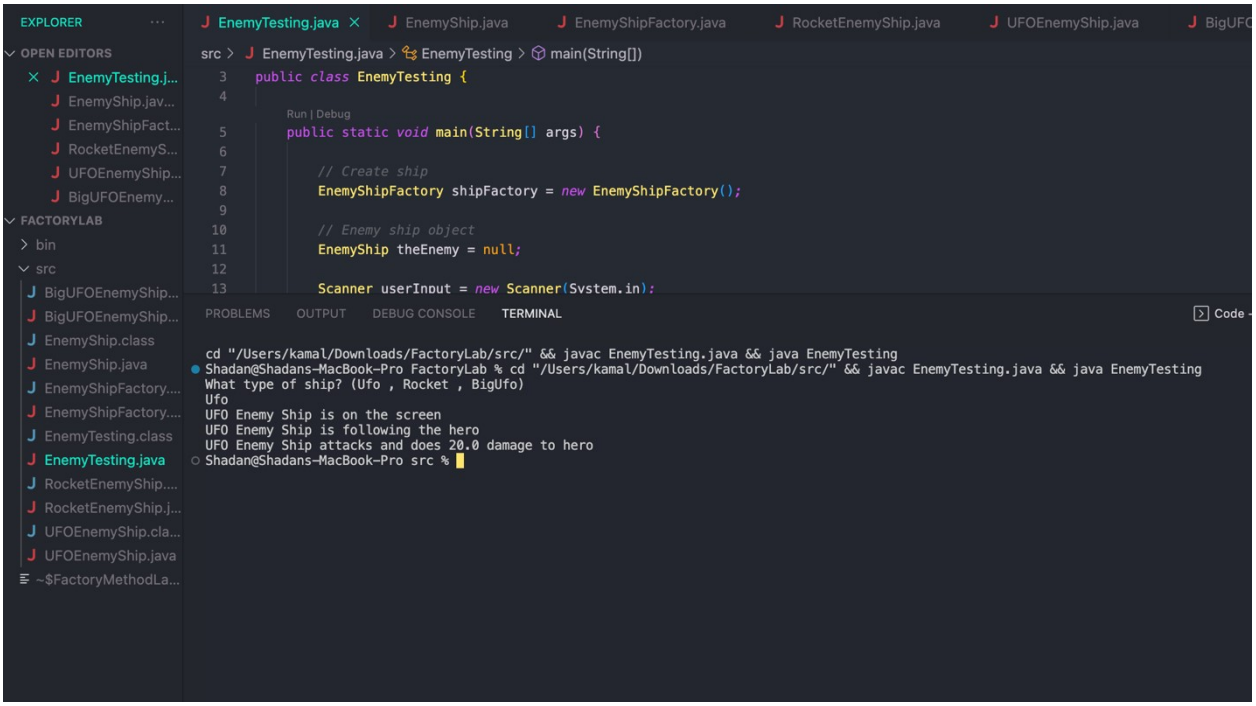

EXPLORER
EnemyTesting.java
EnemyShip.java
EnemyShipFactory.java
RocketEnemyShip.java
UFOEnemyShip.java
BigUFOEnemyShip.java
src > EnemyShipFactory.java > EnemyShipFactory > getShip(String)
public class EnemyShipFactory {
public EnemyShip getShip(String shipType) {
if (shipType == null) {
return null;
}
if (shipType.equalsIgnoreCase(anotherString: "ROCKET")) {
return new RocketEnemyShip();
} else if (shipType.equalsIgnoreCase(anotherString: "UFO")) {
return new UFOEnemyShip();
} else if (shipType.equalsIgnoreCase(anotherString: "BIGUFO")) {
return new BigUFOEnemyShip();
return null;
public void followHeroShip() {
public void displayEnemyShip() {
public void enemyShipShoots() {
The git repository at '/Users/kam


EXPLORER
EnemyTesting.java
EnemyShip.java
EnemyShipFactory.java
RocketEnemyShip.java
UFOEnemyShip.java
BigUFOEnemyShip.java
src > RocketEnemyShip.java > ...
public class RocketEnemyShip extends EnemyShip {
public RocketEnemyShip(){
setName(newName: "Rocket Enemy Ship");
setDamage(newDamage: 10.0);

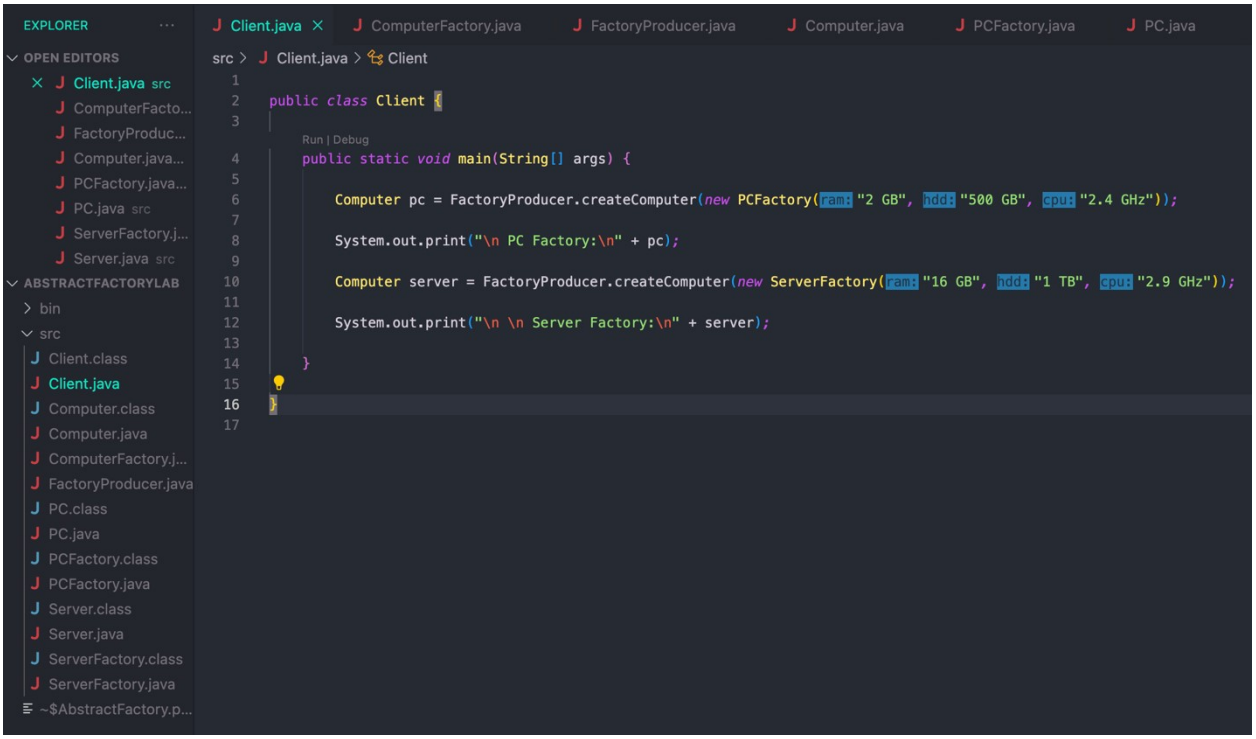

EXPLORER
OPEN EDITORS
EnemyTesting.j...
EnemyShip.jav...
EnemyShipFact...
RocketEnemyS...
UFOEnemyShip...
BigUFOEnemy...
FACTORYLAB
bin
src
BigUFOEnemyShip...
BigUFOEnemyShip...
EnemyShip.class
EnemyShip.java
EnemyShipFactory....
EnemyShipFactory....
EnemyTesting.class
EnemyTesting.java
RocketEnemyShip....
RocketEnemyShip.j...
UFOEnemyShip.cla...
UFOEnemyShip.java
~$FactoryMethodLa...
EnemyTesting.java
EnemyShip.java
EnemyShipFactory.java
RocketEnemyShip.java
UFOEnemyShip.java
BigUFOEnemyShip.java
src > UFOEnemyShip.java > ...
public class UFOEnemyShip extends EnemyShip{
public UFOEnemyShip(){
setName(newName: "UFO Enemy Ship");
setDamage(newDamage: 20.0);
}
}

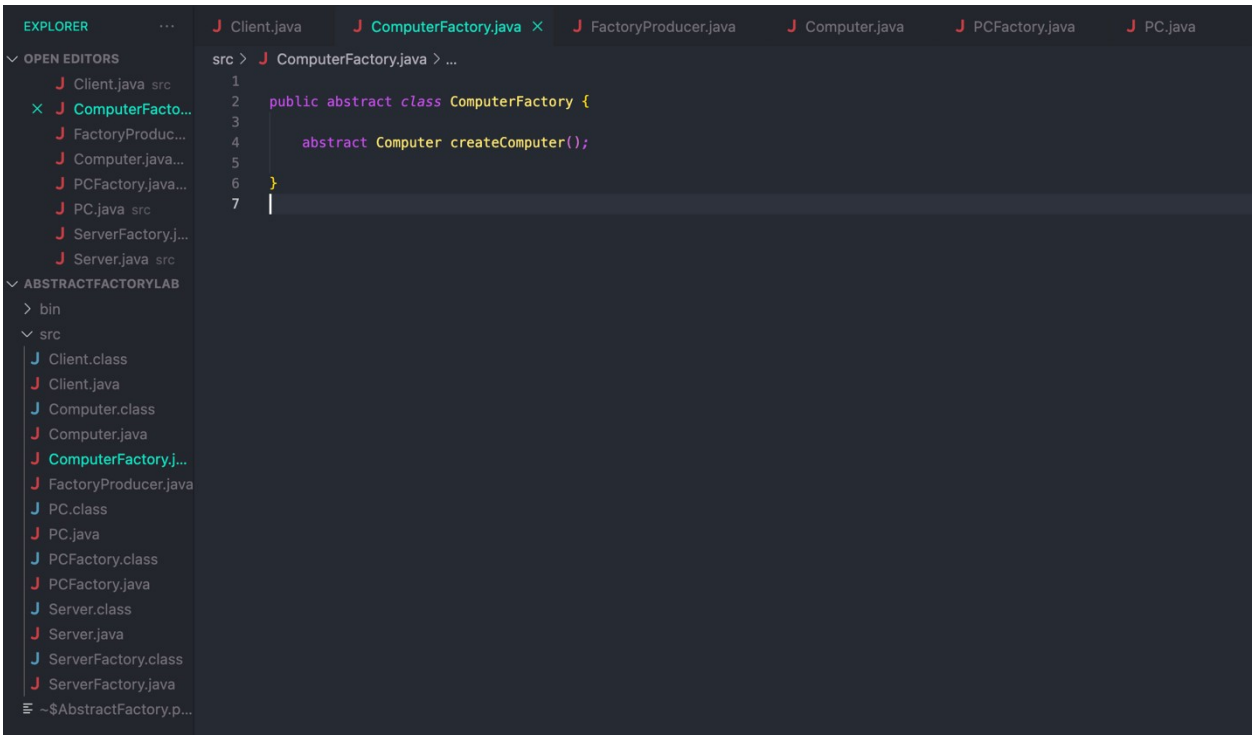

EXPLORER
OPEN EDITORS
EnemyTesting.j...
EnemyShip.jav...
EnemyShipFact...
RocketEnemyS...
UFOEnemyShip...
BigUFOEnemy...
FACTORYLAB
bin
src
BigUFOEnemyShip...
BigUFOEnemyShip...
EnemyShip.class
EnemyShip.java
EnemyShipFactory....
EnemyShipFactory....
EnemyTesting.class
EnemyTesting.java
RocketEnemyShip....
RocketEnemyShip.j...
UFOEnemyShip.cla...
UFOEnemyShip.java
~$FactoryMethodLa...
EnemyTesting.java
EnemyShip.java
EnemyShipFactory.java
RocketEnemyShip.java
UFOEnemyShip.java
BigUFOEnemyShip.java
src > BigUFOEnemyShip.java > ...
public class BigUFOEnemyShip extends UFOEnemyShip{
public BigUFOEnemyShip(){
setName(newName: "Big UFO Enemy Ship");
setDamage(newDamage: 40.0);
}
}

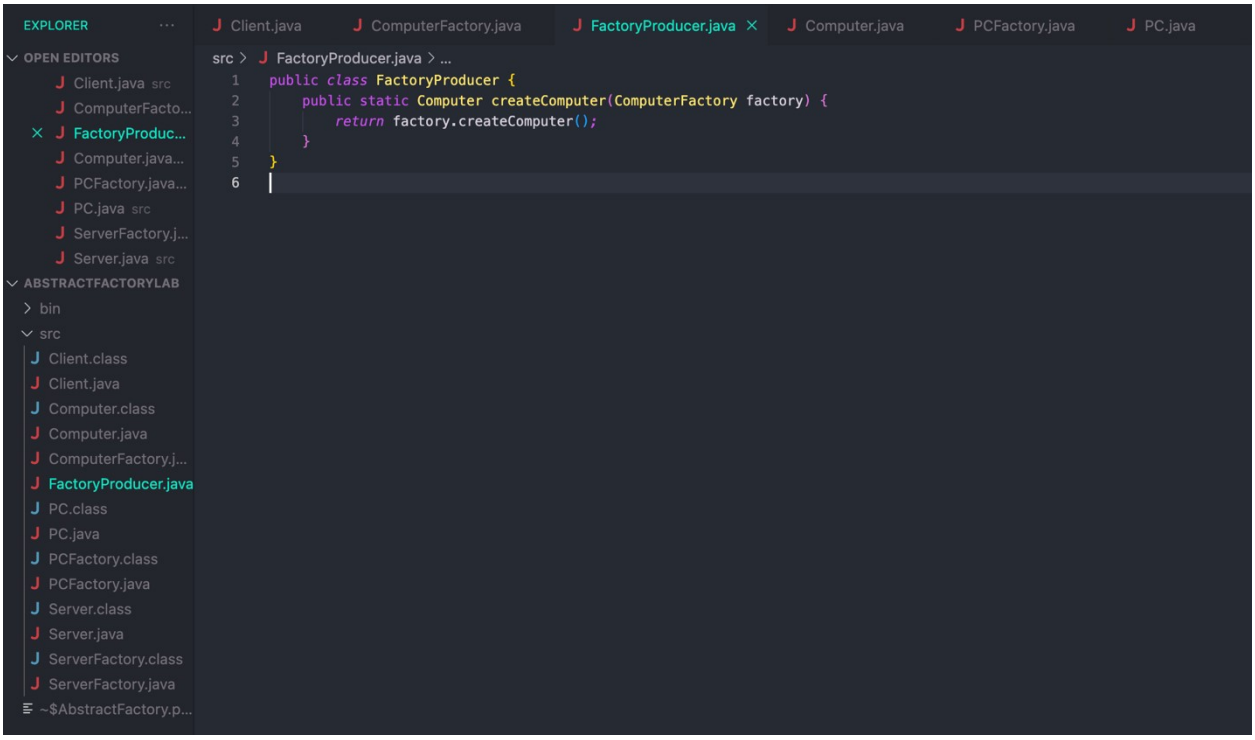
Output:

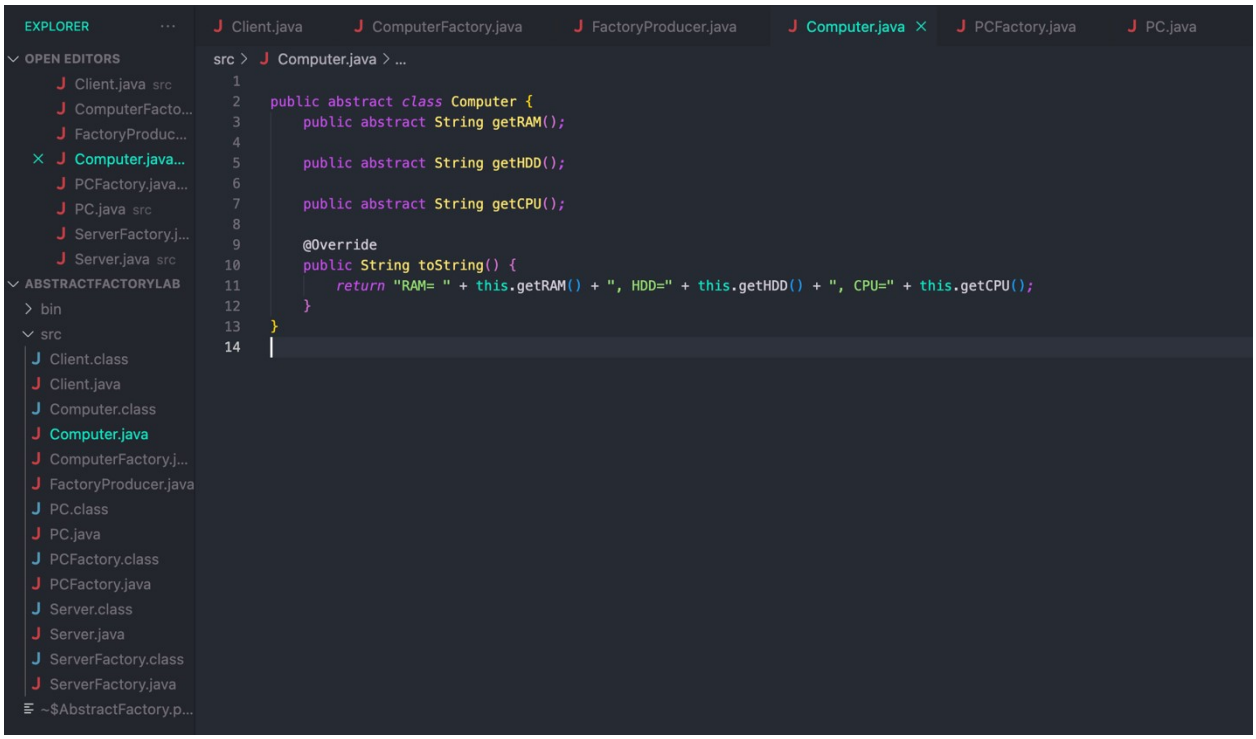


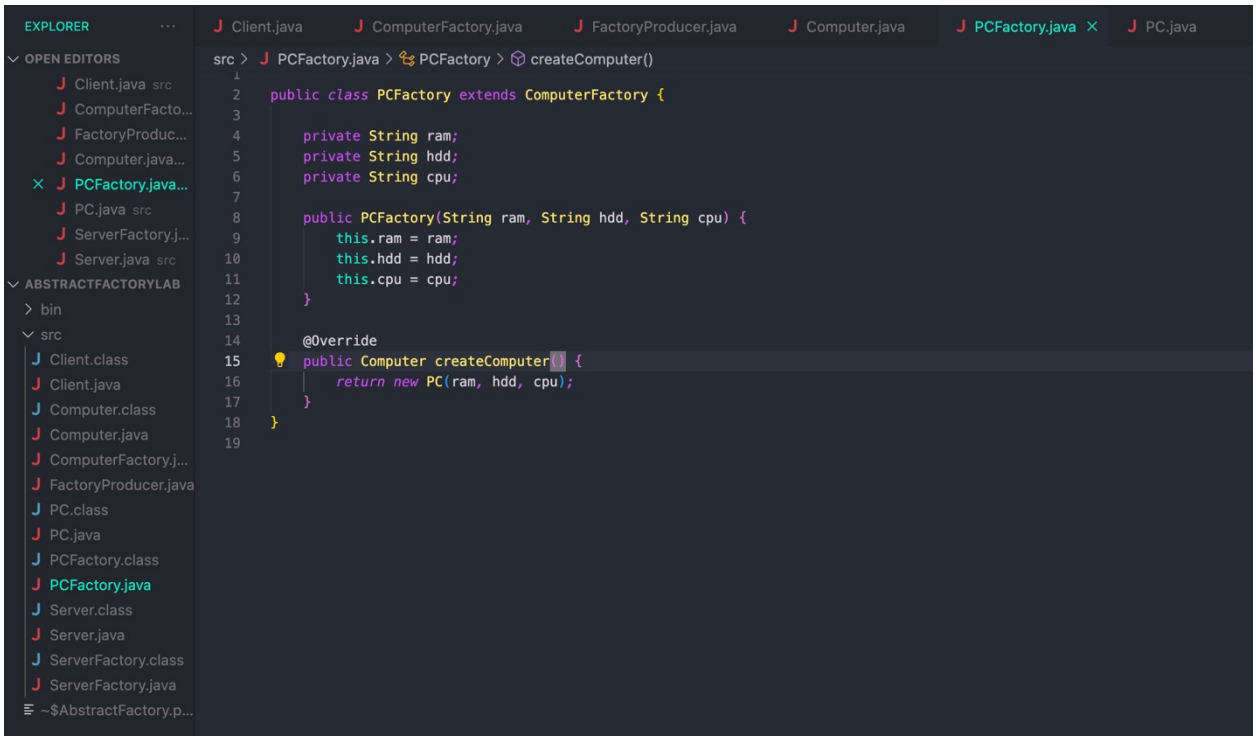
Abstract pattern:

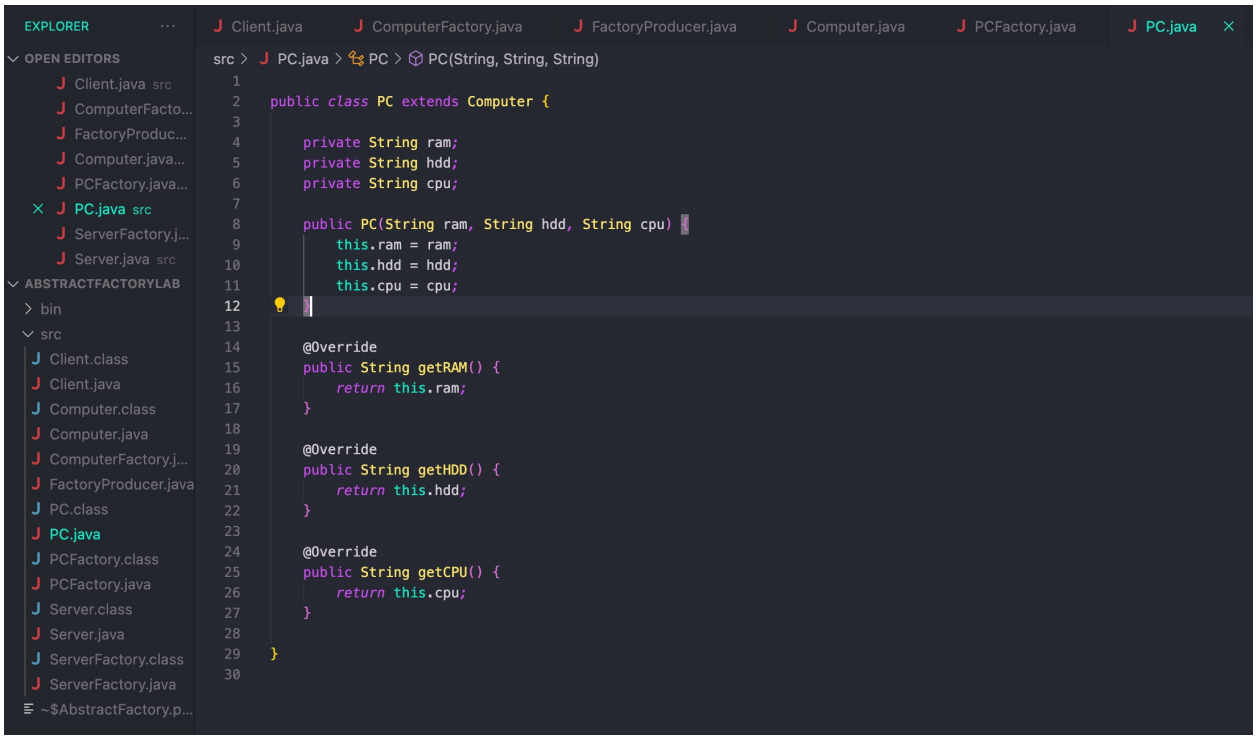


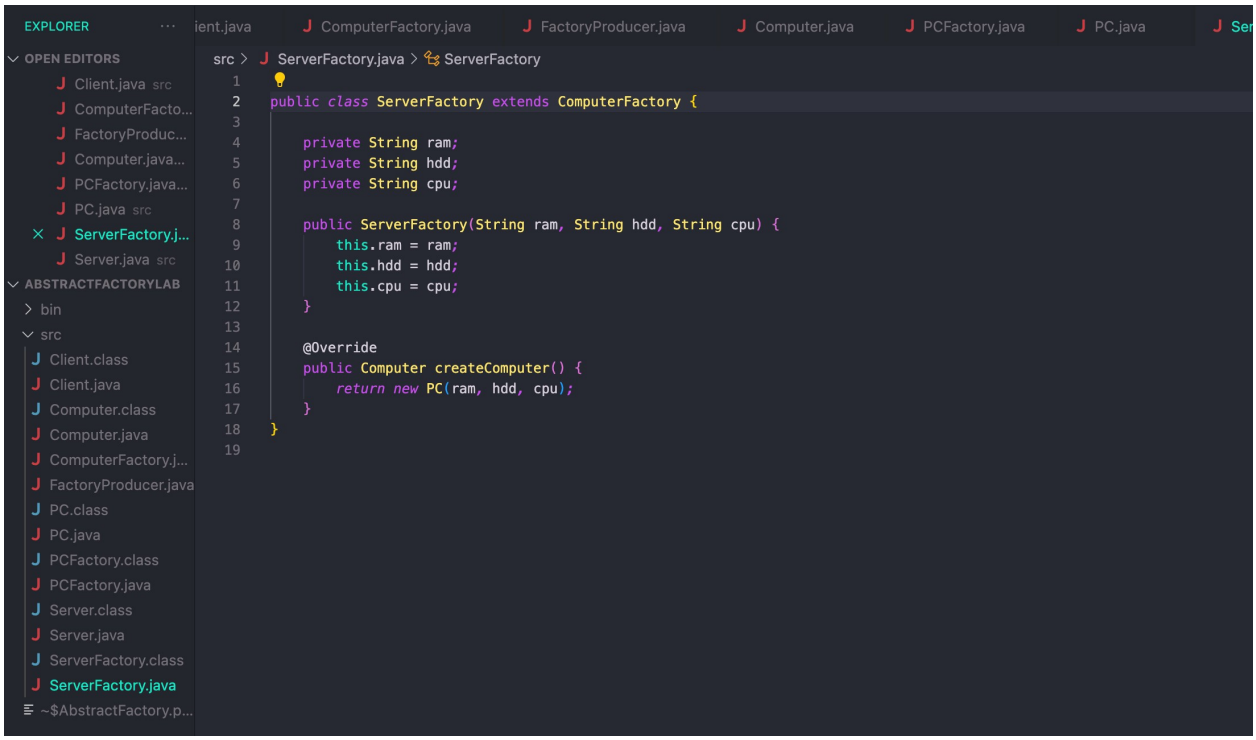

EXPLORER
Client.java
ComputerFactory.java
FactoryProducer.java
Computer.java
PCFactory.java
PC.java
src > ComputerFactory.java > ...
1
2 public abstract class ComputerFactory {
3
4 abstract Computer createComputer();
5
6 }
7
OPEN EDITORS
Client.java src
ComputerFacto...
FactoryProduc...
Computer.java...
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
~$AbstractFactory.p...

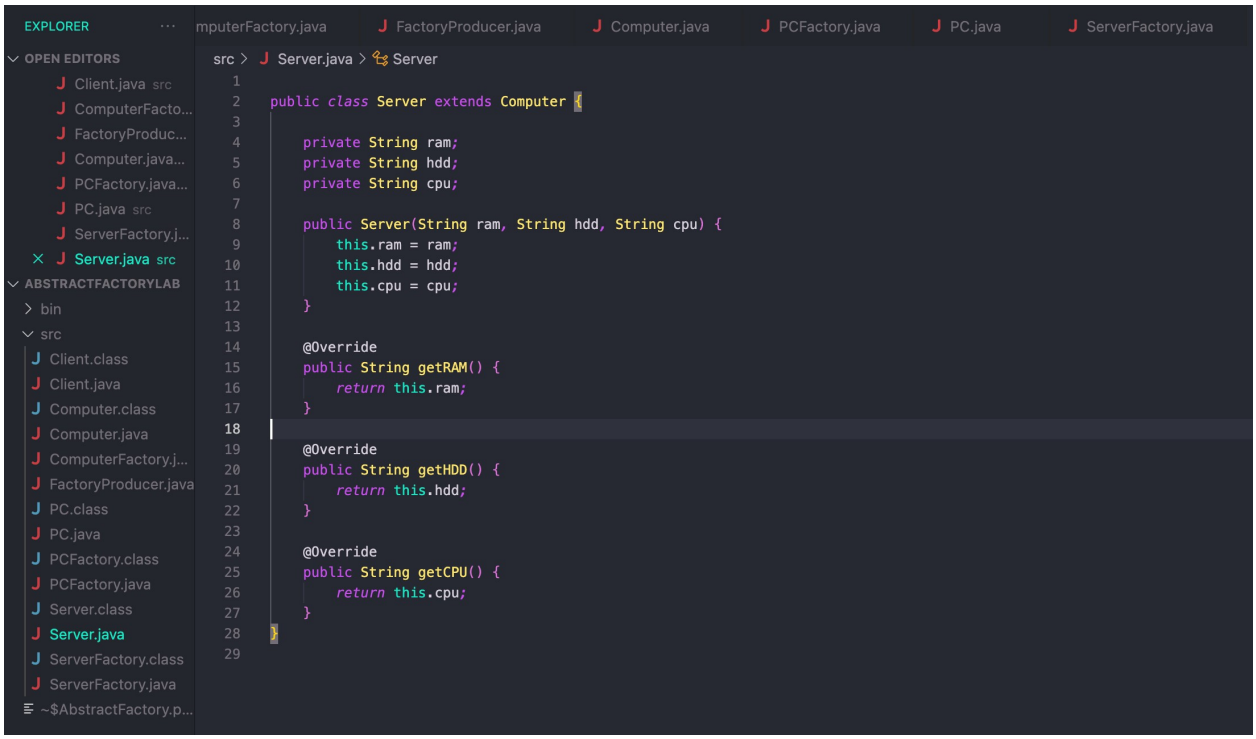

EXPLORER
Client.java
ComputerFactory.java
FactoryProducer.java
Computer.java
PCFactory.java
PC.java
src > FactoryProducer.java > ...
1 public class FactoryProducer {
2 public static Computer createComputer(ComputerFactory factory) {
3 return factory.createComputer();
4 }
5 }
6
OPEN EDITORS
Client.java 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
~$AbstractFactory.p...


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
~$AbstractFactory.p...
Client.java
ComputerFactory.java
FactoryProducer.java
Computer.java
PCFactory.java
PC.java
src > Computer.java > ...
1
public abstract class Computer {
public abstract String getRAM();
public abstract String getHDD();
public abstract String getCPU();
@Override
public String toString() {
return "RAM= " + this.getRAM() + ", HDD=" + this.getHDD() + ", CPU=" + this.getCPU();
}
}
14


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
~$AbstractFactory.p...
Client.java
ComputerFactory.java
FactoryProducer.java
Computer.java
PCFactory.java
PC.java
src > PCFactory.java > PCFactory > createComputer()
public class PCFactory extends ComputerFactory {
private String ram;
private String hdd;
private String cpu;
public PCFactory(String ram, String hdd, String cpu) {
this.ram = ram;
this.hdd = hdd;
this.cpu = cpu;
}
@Override
public Computer createComputer() {
return new PC(ram, hdd, cpu);
}
}


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
~$AbstractFactory.p...
Client.java
ComputerFactory.java
FactoryProducer.java
Computer.java
PCFactory.java
PC.java
src > PC.java > PC > PC(String, String, String)
public class PC extends Computer {
private String ram;
private String hdd;
private String cpu;
public PC(String ram, String hdd, String cpu) {
this.ram = ram;
this.hdd = hdd;
this.cpu = cpu;
}
@Override
public String getRAM() {
return this.ram;
}
@Override
public String getHDD() {
return this.hdd;
}
@Override
public String getCPU() {
return this.cpu;
}
}


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
~$AbstractFactory.p...
ComputerFactory.java
FactoryProducer.java
Computer.java
PCFactory.java
PC.java
src > ServerFactory.java > ServerFactory
public class ServerFactory extends ComputerFactory {
private String ram;
private String hdd;
private String cpu;
public ServerFactory(String ram, String hdd, String cpu) {
this.ram = ram;
this.hdd = hdd;
this.cpu = cpu;
}
@Override
public Computer createComputer() {
return new PC(ram, hdd, cpu);
}
}


EXPLORER
mputerFactory.java
FactoryProducer.java
Computer.java
PCFactory.java
PC.java
ServerFactory.java
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
~$AbstractFactory.p...
src > Server.java > Server
public class Server extends Computer {
private String ram;
private String hdd;
private String cpu;
public Server(String ram, String hdd, String cpu) {
this.ram = ram;
this.hdd = hdd;
this.cpu = cpu;
}
@Override
public String getRAM() {
return this.ram;
}
@Override
public String getHDD() {
return this.hdd;
}
@Override
public String getCPU() {
return this.cpu;
}
}

Output:


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.c...
ComputerFactory.j...
FactoryProducer.cl...
FactoryProducer.java
PC.class
PC.java
PCFactory.class
PCFactory.java
Server.class
Server.java
ServerFactory.class
ServerFactory.java
~$AbstractFactory.p...
TIMELINE
public class Client {
Run | Debug
public static void main(String[] args) {
Computer pc = FactoryProducer.createComputer(new PCFactory(ram: "2 GB", hdd: "500 GB", cpu: "2.4 GHz"));
System.out.print("\n PC Factory:\n" + pc);
Computer server = FactoryProducer.createComputer(new ServerFactory(ram: "16 GB", hdd: "1 TB", cpu: "2.9 GHz"));
System.out.print("\n \n Server Factory:\n" + server);
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code - s
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 %