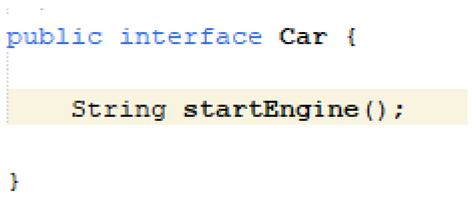
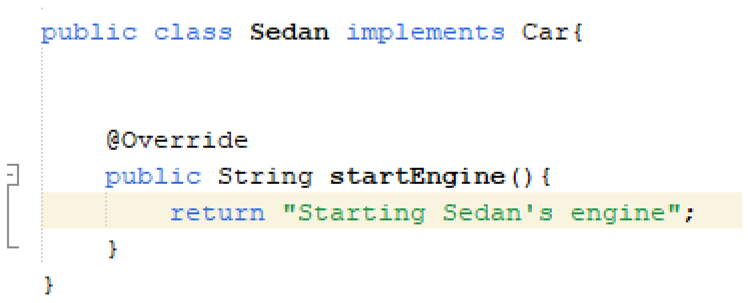
Laboratorium 5



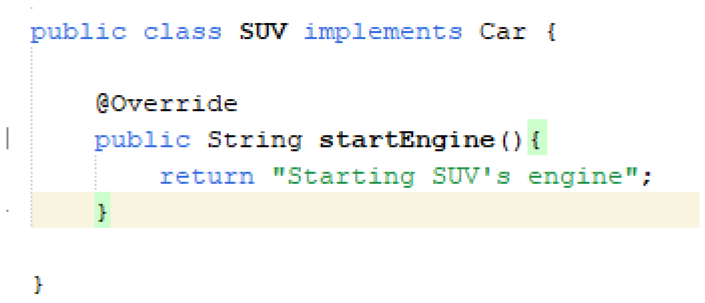
```java
public interface Car {

    String startEngine();

}
```

Rys 1 – interfejs Car



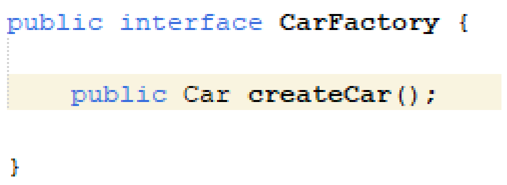
```java
public class Sedan implements Car{


    @Override
    public String startEngine(){
        return "Starting Sedan's engine";
    }
}
```

Rys. 2 – klasa Sedan



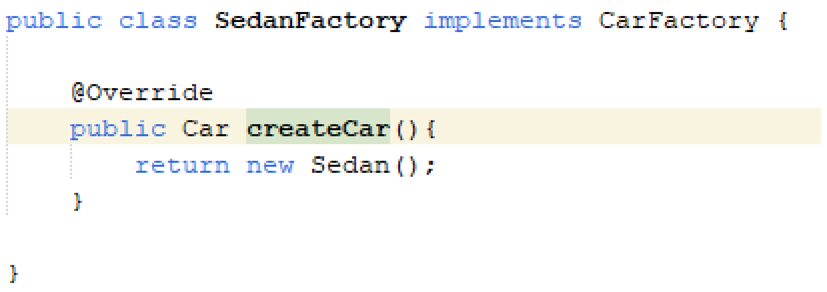
```java
public class SUV implements Car {

    @Override
    public String startEngine(){
        return "Starting SUV's engine";
    }

}
```

Rys. 3 – klasa SUV



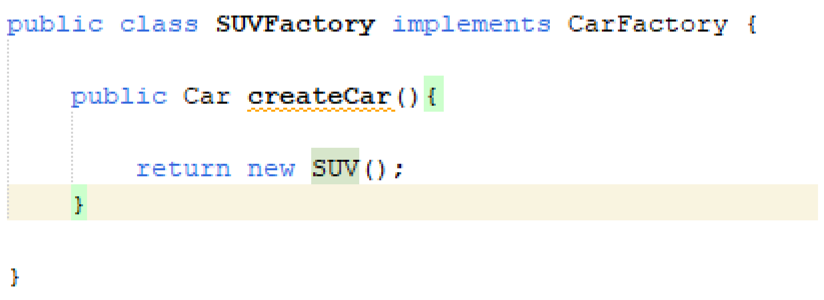
```java
public interface CarFactory {

    public Car createCar();

}
```

Rys. 4 – interfejs CarFactory



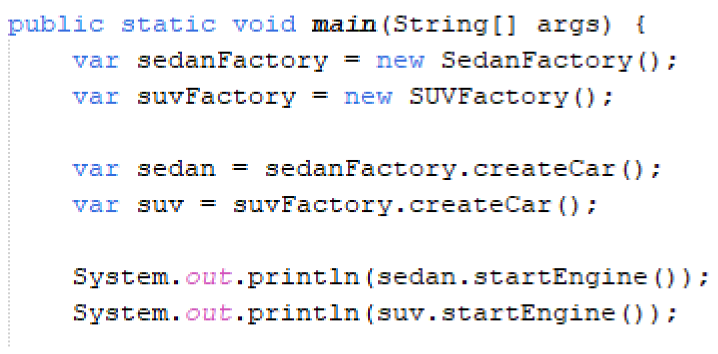
```java
public class SedanFactory implements CarFactory {

    @Override
    public Car createCar(){
        return new Sedan();
    }

}
```

Rys. 5 – klasa SedanFactory



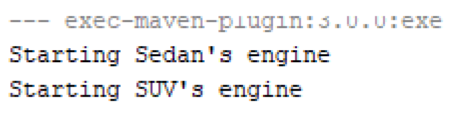
```java
public class SUVFactory implements CarFactory {

    public Car createCar(){

        return new SUV();
    }

}
```

Rys. 6 – klasa SUVFactory



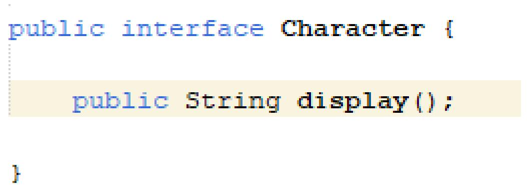
```java
public static void main(String[] args) {
    var sedanFactory = new SedanFactory();
    var suvFactory = new SUVFactory();

    var sedan = sedanFactory.createCar();
    var suv = suvFactory.createCar();

    System.out.println(sedan.startEngine());
    System.out.println(suv.startEngine());
```

Rys. 7 – metoda main()



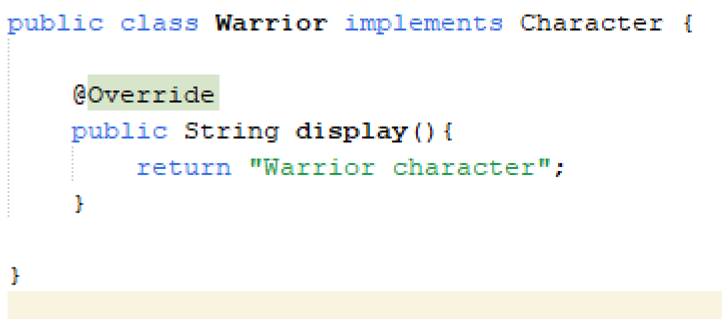
```
--- exec-maven-plugin:3.0.0:exe
Starting Sedan's engine
Starting SUV's engine
```

Rys. 8 – rezultat działania metody main()



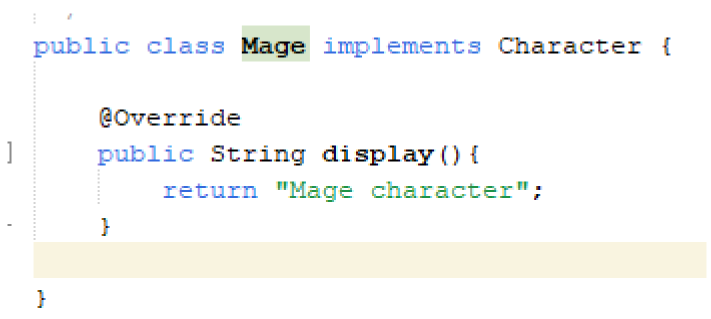
```java
public interface Character {

    public String display();

}
```

Rys. 9 – interfejs Character



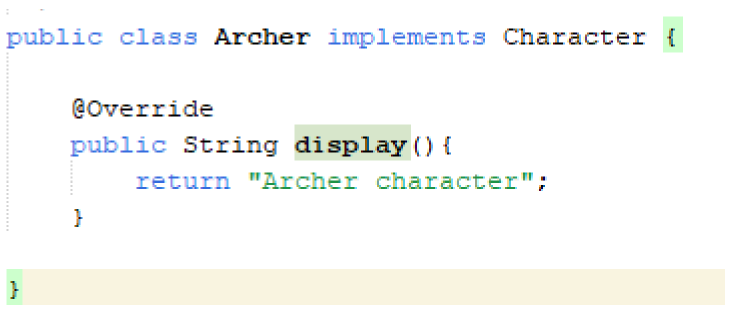
```java
public class Warrior implements Character {

    @Override
    public String display(){
        return "Warrior character";
    }

}
```

Rys. 10 – klasa Warrior



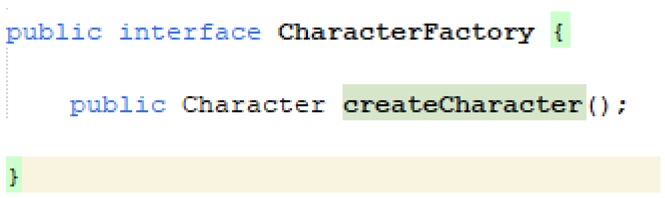
```java
public class Mage implements Character {

    @Override
    public String display(){
        return "Mage character";
    }

}
```

Rys. 11 – klasa Mage



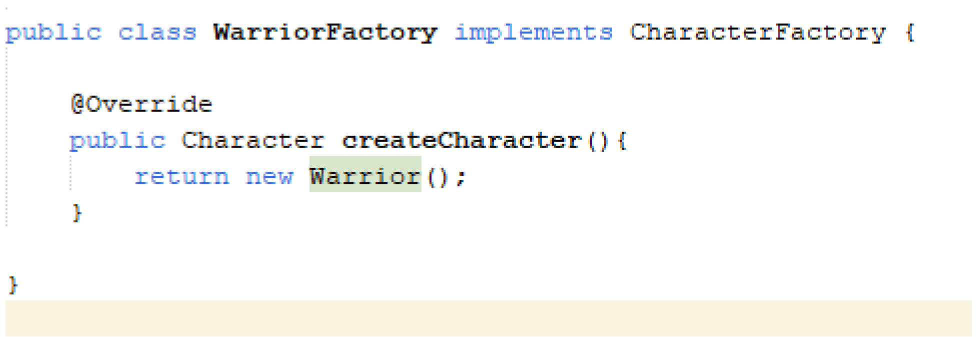
```java
public class Archer implements Character {

    @Override
    public String display(){
        return "Archer character";
    }

}
```

Rys. 12 – klasa Archer



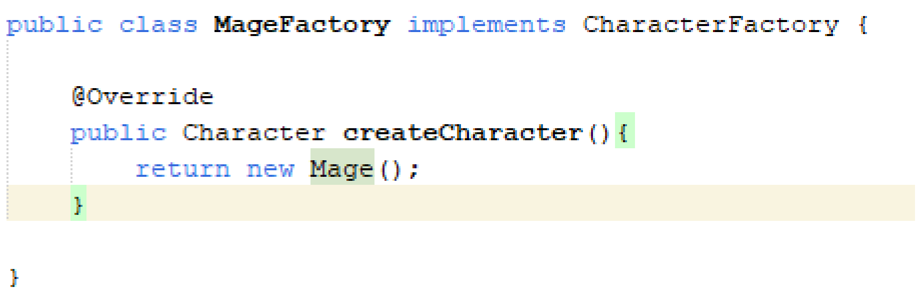
```java
public interface CharacterFactory {

    public Character createCharacter();

}
```

Rys. 13 – interfejs CharacterFactory



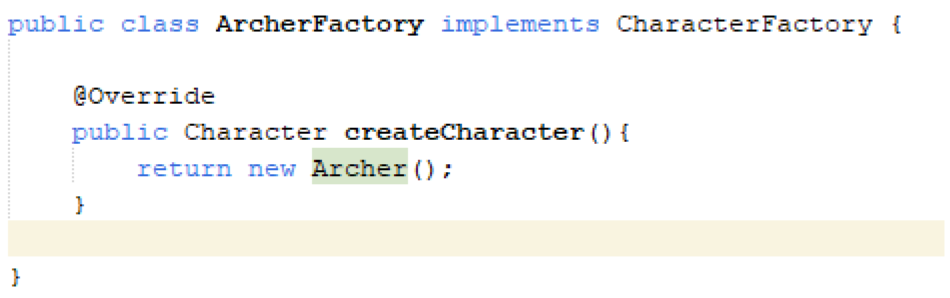
```java
public class WarriorFactory implements CharacterFactory {

    @Override
    public Character createCharacter(){
        return new Warrior();
    }

}
```

Rys. 14 – klasa WarriorFactory



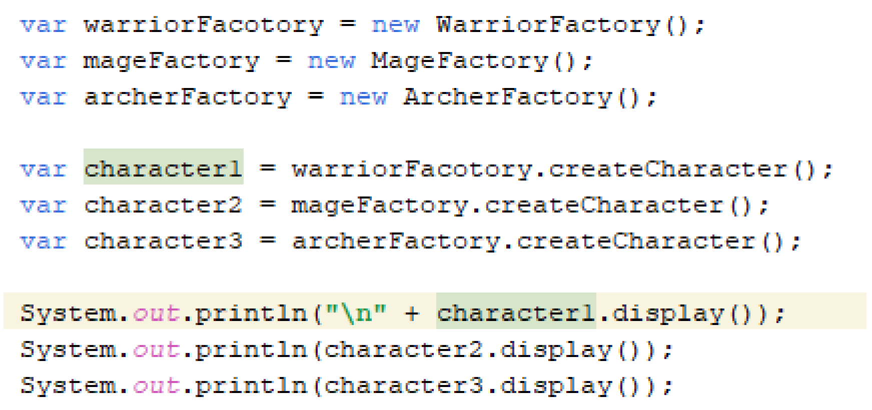
```java
public class MageFactory implements CharacterFactory {

    @Override
    public Character createCharacter(){
        return new Mage();
    }

}
```

Rys. 15 – klasa MageFactory



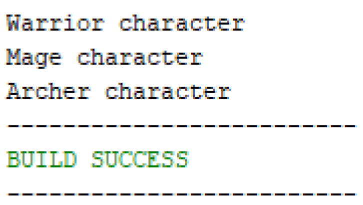
```java
public class ArcherFactory implements CharacterFactory {

    @Override
    public Character createCharacter(){
        return new Archer();
    }

}
```

Rys. 16 – klasa ArcherFactory



```java
var warriorFacotory = new WarriorFactory();
var mageFactory = new MageFactory();
var archerFactory = new ArcherFactory();

var character1 = warriorFacotory.createCharacter();
var character2 = mageFactory.createCharacter();
var character3 = archerFactory.createCharacter();

System.out.println("\n" + character1.display());
System.out.println(character2.display());
System.out.println(character3.display());
```

Rys. 17 – metoda main()



```
Warrior character
Mage character
Archer character
------------------------
BUILD SUCCESS
------------------------
```

Rys. 18 – rezultat działania metody main()