How do you reverse the string in java? Illustrate with example.

```
~

public class main{
    public static void main(String[] args) {
        System.out.println(new StringBuilder("Something").reverse().toString());
    }
}
```

```
~

[wizard@archlinux 5]$ java main
gnihtemoS
[wizard@archlinux 5]$
```

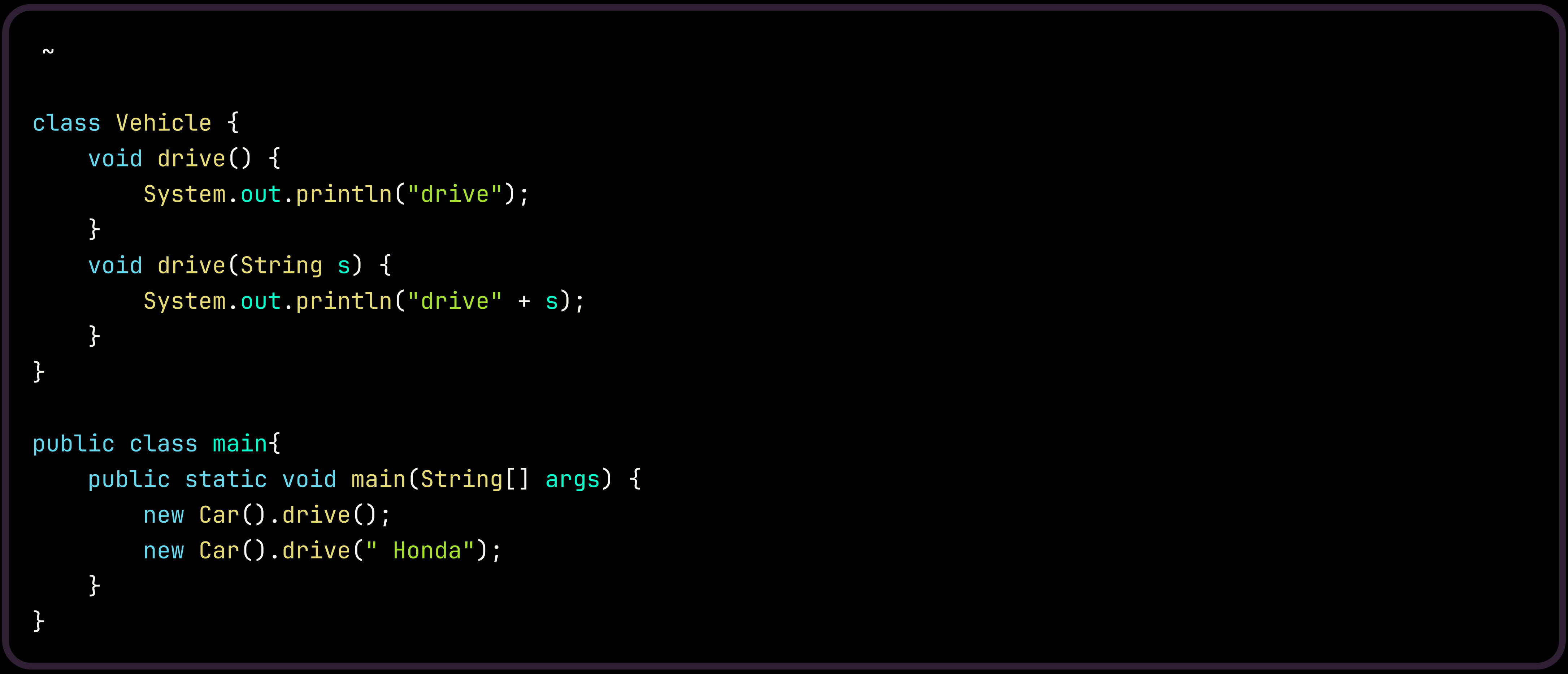
Write a Java program to create a class called Vehicle with a method called drive(). Create a subclass called Car that overrides the drive() method to print "Repairing a car".

```
~

class Vehicle {
    void drive() {
        System.out.println("drive");
    }
}

class Car extends Vehicle{
    void drive(){
        System.out.println("Reparing car");
    }
}
public class main{
    public static void main(String[] args) {
        new Car().drive();
    }
}
```

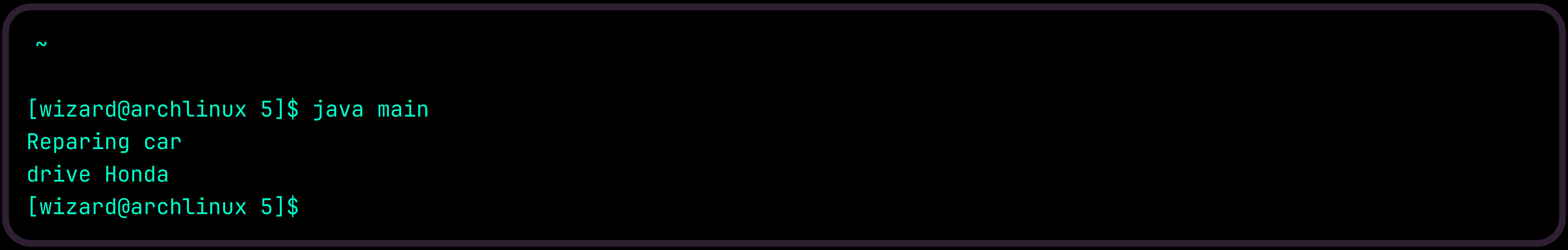
```
~

[wizard@archlinux 5]$ java main
Reparing car
[wizard@archlinux 5]$
```

What is method overloading and method overriding? Illustrate with examples.

Method overriding is replacing an existing method from a child, whereas method overloading is extending an existing method to support either different datatypes or different number of parameters. Method Overriding example is above, overloading's is below:

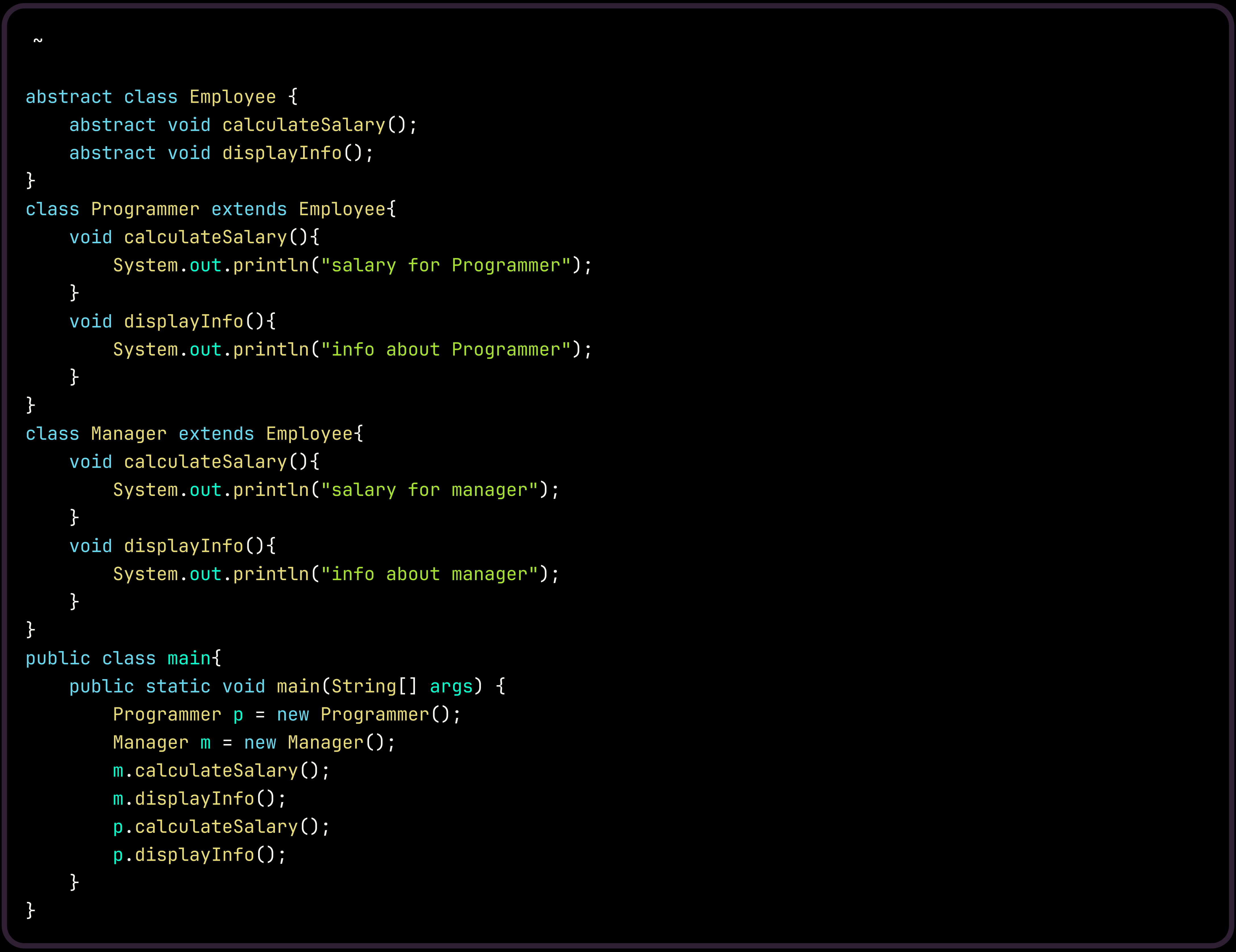


```
~

class Vehicle {
    void drive() {
        System.out.println("drive");
    }
    void drive(String s) {
        System.out.println("drive" + s);
    }
}

public class main{
    public static void main(String[] args) {
        new Car().drive();
        new Car().drive(" Honda");
    }
}
```

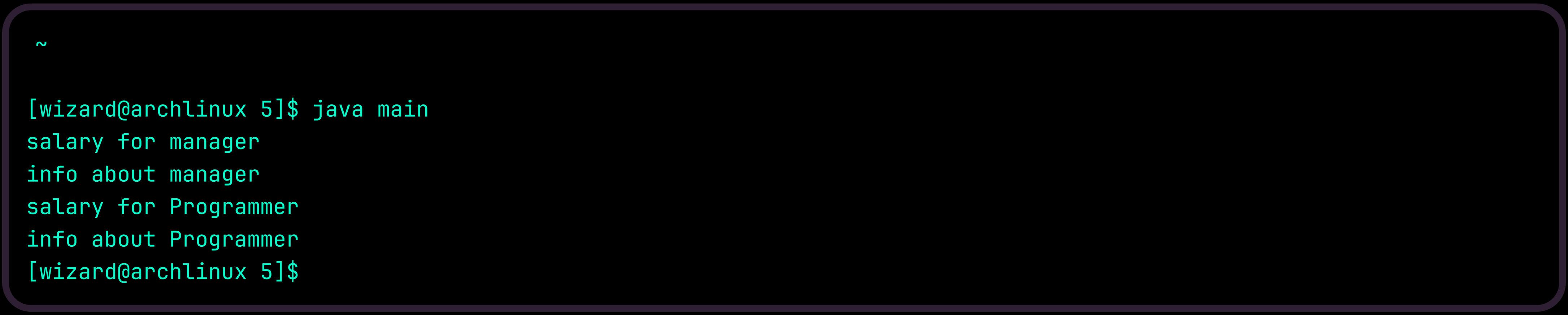


```
~

[wizard@archlinux 5]$ java main
Reparing car
drive Honda
[wizard@archlinux 5]$
```

Write a Java program to create an abstract class Employee with abstract methods calculateSalary() and displayInfo(). Create subclasses Manager and Programmer that extend the Employee class and implement the respective methods to calculate salary and display information for each role.

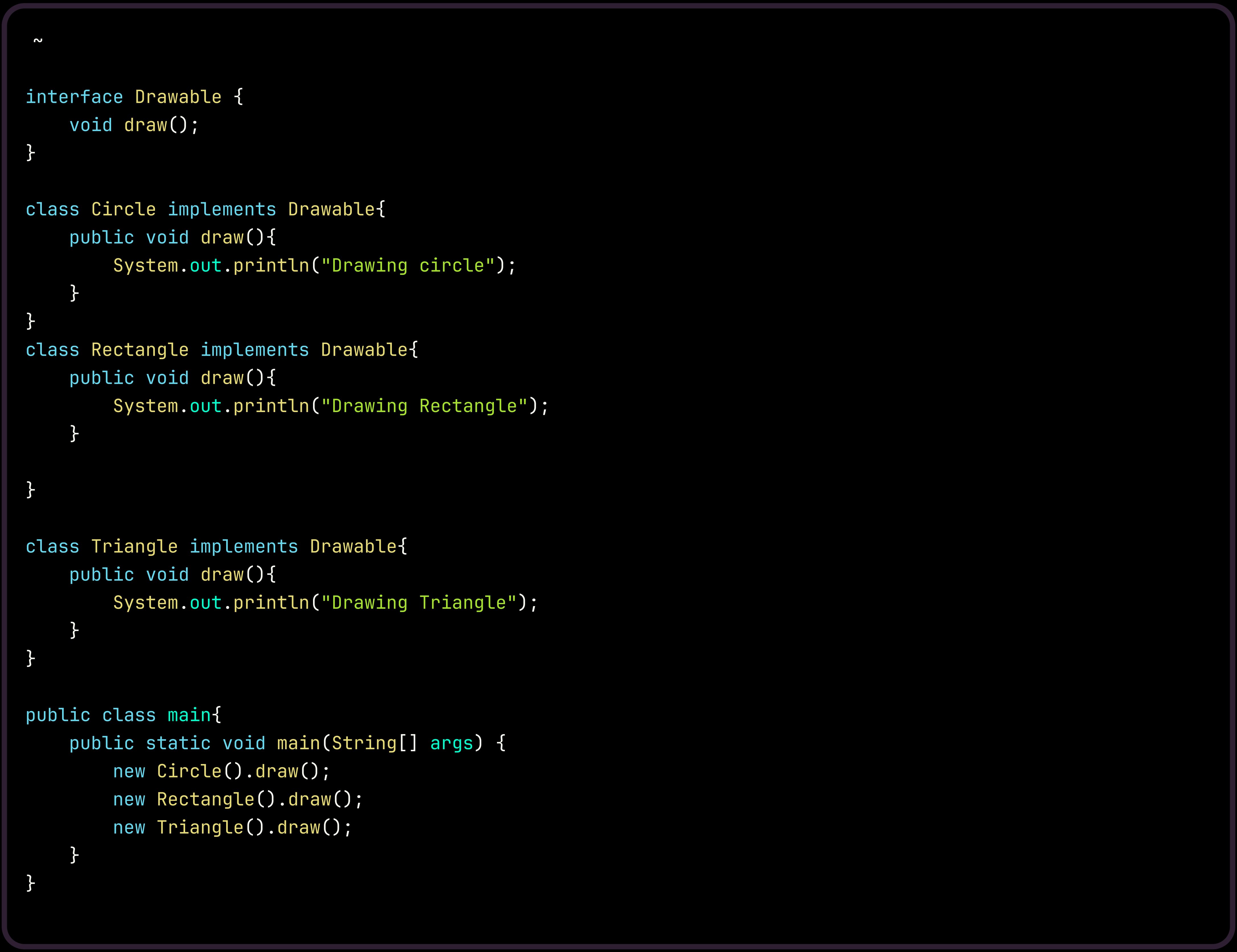


```
~

abstract class Employee {
    abstract void calculateSalary();
    abstract void displayInfo();
}
class Programmer extends Employee{
    void calculateSalary(){
        System.out.println("salary for Programmer");
    }
    void displayInfo(){
        System.out.println("info about Programmer");
    }
}
class Manager extends Employee{
    void calculateSalary(){
        System.out.println("salary for manager");
    }
    void displayInfo(){
        System.out.println("info about manager");
    }
}
public class main{
    public static void main(String[] args) {
        Programmer p = new Programmer();
        Manager m = new Manager();
        m.calculateSalary();
        m.displayInfo();
        p.calculateSalary();
        p.displayInfo();
    }
}
```

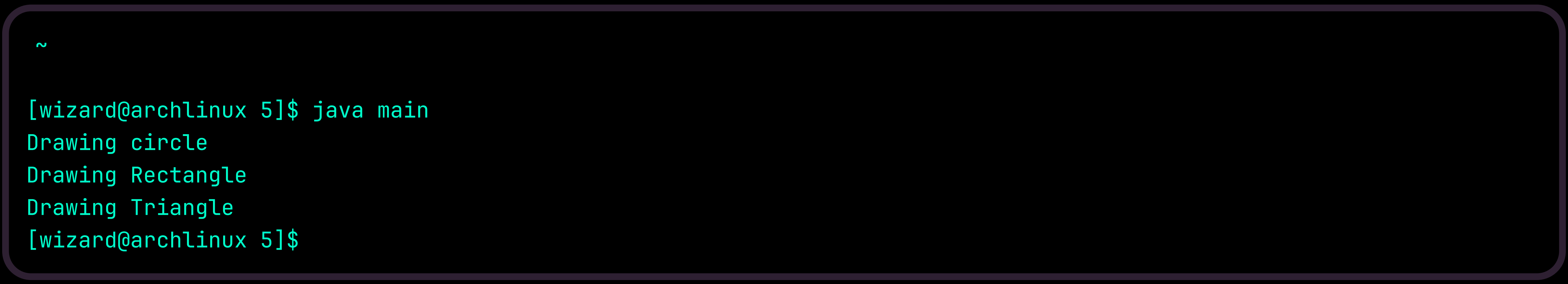


```
~

[wizard@archlinux 5]$ java main
salary for manager
info about manager
salary for Programmer
info about Programmer
[wizard@archlinux 5]$
```

Write a Java program to create an interface Drawable with a method draw() that takes no arguments and returns void. Create three classes Circle, Rectangle, and Triangle that implement the Drawable interface and override the draw() method to draw their respective shapes.

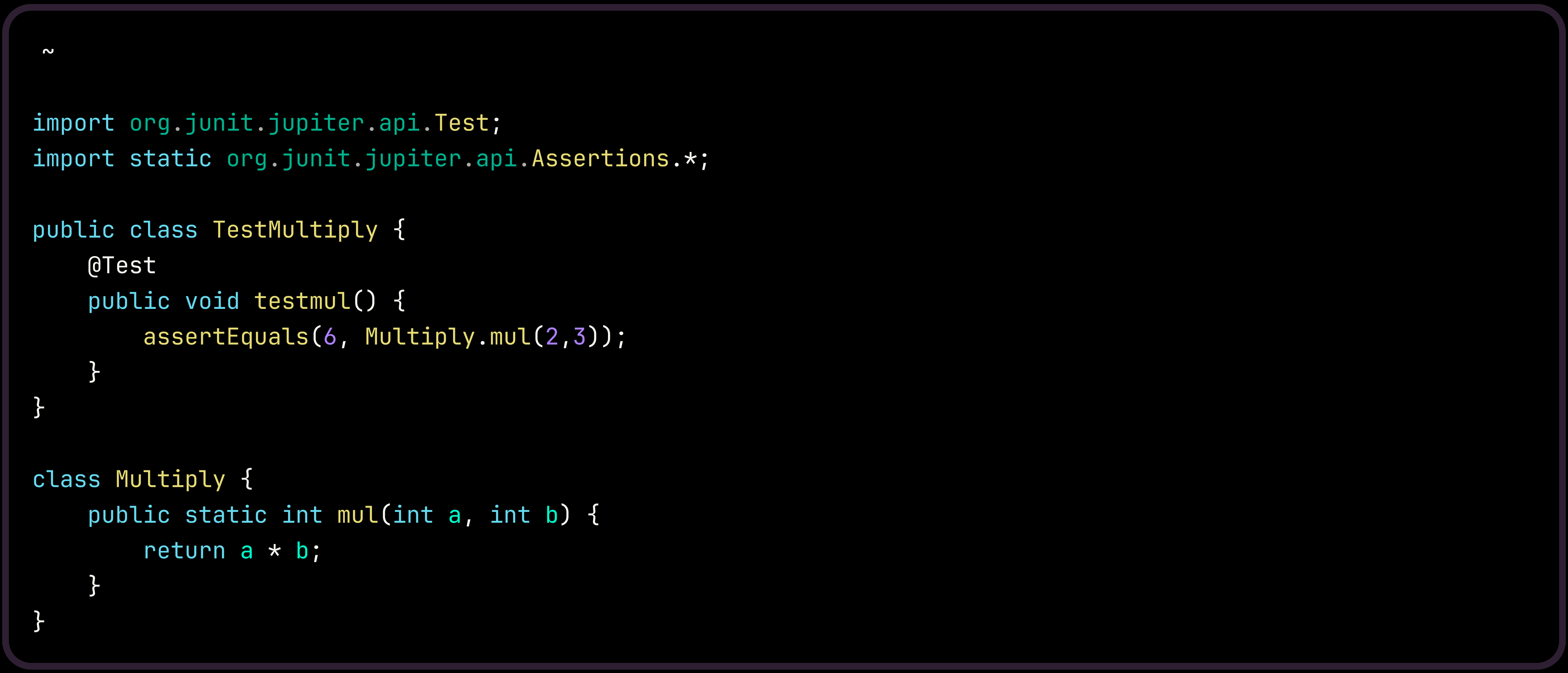


```
~

interface Drawable {
    void draw();
}

class Circle implements Drawable{
    public void draw(){
        System.out.println("Drawing circle");
    }
}
class Rectangle implements Drawable{
    public void draw(){
        System.out.println("Drawing Rectangle");
    }

}

class Triangle implements Drawable{
    public void draw(){
        System.out.println("Drawing Triangle");
    }
}

public class main{
    public static void main(String[] args) {
        new Circle().draw();
        new Rectangle().draw();
        new Triangle().draw();
    }
}
```

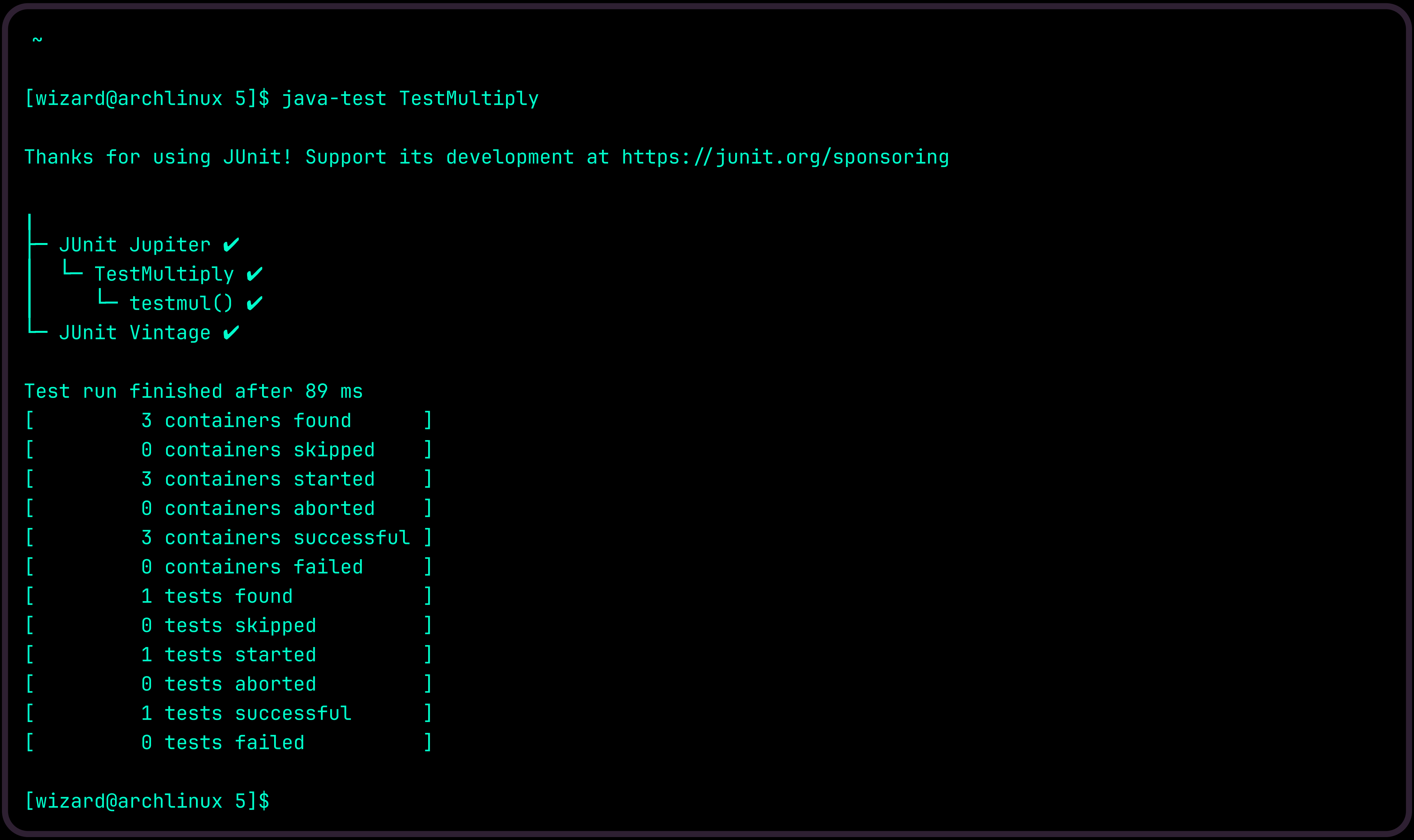


```
~

[wizard@archlinux 5]$ java main
Drawing circle
Drawing Rectangle
Drawing Triangle
[wizard@archlinux 5]$
```

Write a junit test to add, subtract and multiply two numbers.



```java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class TestMultiply {
    @Test
    public void testmul() {
        assertEquals(6, Multiply.mul(2,3));
    }
}

class Multiply {
    public static int mul(int a, int b) {
        return a * b;
    }
}
```



```
[wizard@archlinux 5]$ java-test TestMultiply

Thanks for using JUnit! Support its development at https://junit.org/sponsoring

│
├─ JUnit Jupiter ✔
│  └─ TestMultiply ✔
│     └─ testmul() ✔
└─ JUnit Vintage ✔

Test run finished after 89 ms
[         3 containers found      ]
[         0 containers skipped    ]
[         3 containers started    ]
[         0 containers aborted    ]
[         3 containers successful ]
[         0 containers failed     ]
[         1 tests found           ]
[         0 tests skipped         ]
[         1 tests started         ]
[         0 tests aborted         ]
[         1 tests successful      ]
[         0 tests failed          ]

[wizard@archlinux 5]$
```