Solid Principles Assignment

Aidrin Varghese

ILP Batch 2

23 / 01 / 2023

Single Responsibility Principle

Customer Class

The **Customer** class is responsible for encapsulating customer information, providing methods to retrieve the customer's name and email. It adheres to the SRP by focusing solely on the management of customer data.

```java
package com.ilp.entity;



public class Customer {
private String name;
private String email;



public Customer(String name, String email) {
this.name = name;
this.email = email;
}



public String getName() {
return name;
}



public String getEmail() {
return email;
}
}
```

Single Responsibility Principle

Issue Class

The **Issue** class represents an issue within the system, maintaining information about the issue description and its resolution status. It follows the SRP by concentrating on the management of issue data.

```java
package com.ilp.entity;

public class Issue {
private String description;
private boolean resolved;

public Issue(String description) {
this.description = description;
this.resolved = false;
}

public String getDescription() {
return description;
}

public boolean isResolved() {
return resolved;
}

public void resolveIssue(boolean resolved) {
this.resolved = resolved;
}
}
```

Open – Closed Principles

IssueResolver Interface and Implementations

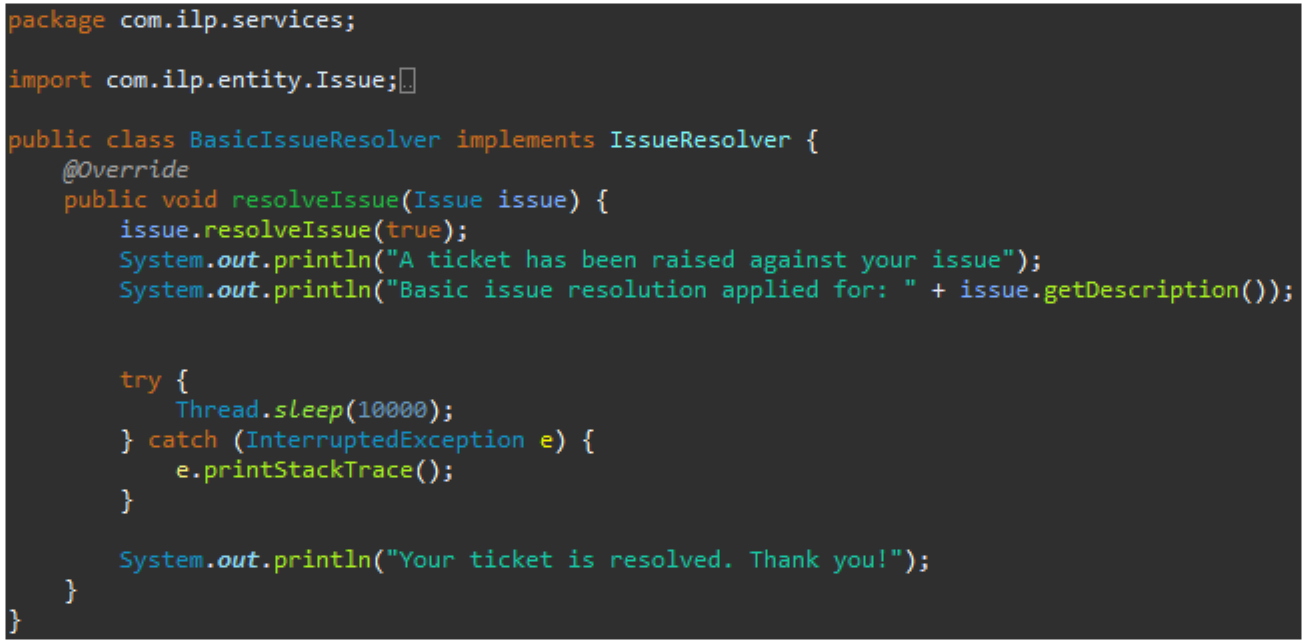
The **IssueResolver** interface is designed to be open for extension, allowing the system to introduce new issue resolution strategies without modifying existing code. Both **BasicIssueResolver** and **AdvancedIssueResolver** demonstrate this extensibility.

```
package com.ilp.interfaces;

import com.ilp.entity.Issue;

//Abstraction through Interface
(IssueResolver)
public interface IssueResolver {
void resolveIssue(Issue issue);
}
```

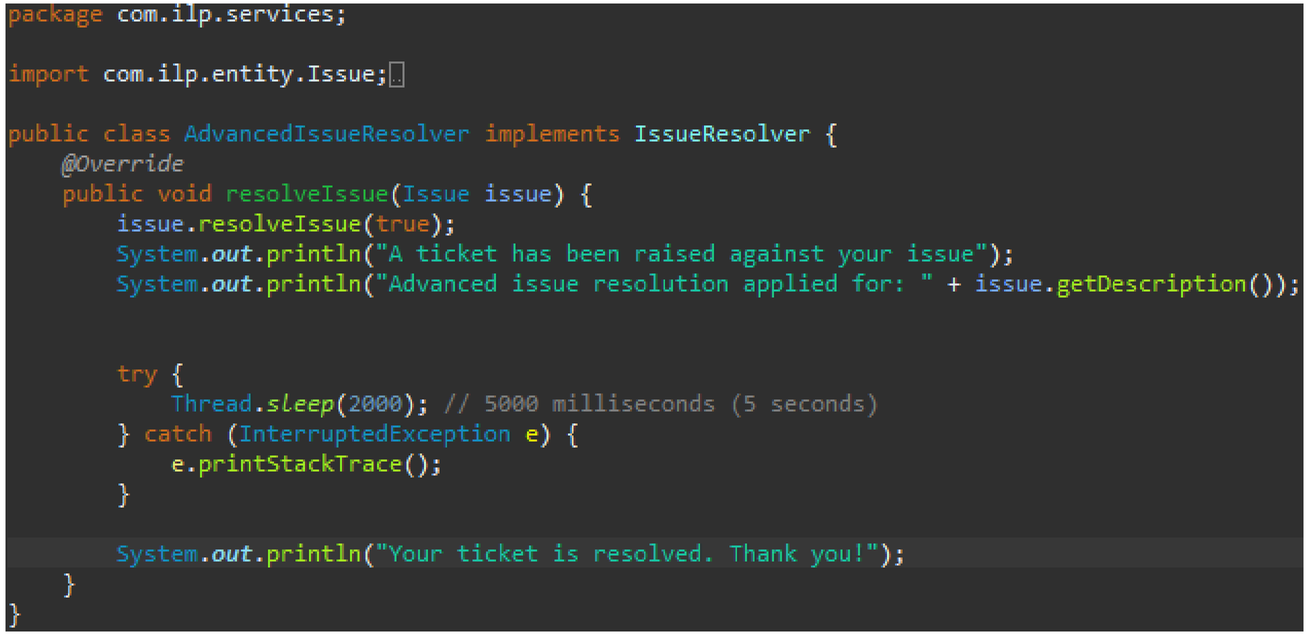
Open – Closed Principles

IssueResolver Interface and Implementations

The **IssueResolver** interface is designed to be open for extension, allowing the system to introduce new issue resolution strategies without modifying existing code. Both **BasicIssueResolver** and **AdvancedIssueResolver** demonstrate this extensibility.



```java
package com.ilp.services;

import com.ilp.entity.Issue;

public class BasicIssueResolver implements IssueResolver {
    @Override
    public void resolveIssue(Issue issue) {
        issue.resolveIssue(true);
        System.out.println("A ticket has been raised against your issue");
        System.out.println("Basic issue resolution applied for: " + issue.getDescription());


        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Your ticket is resolved. Thank you!");
    }
}
```

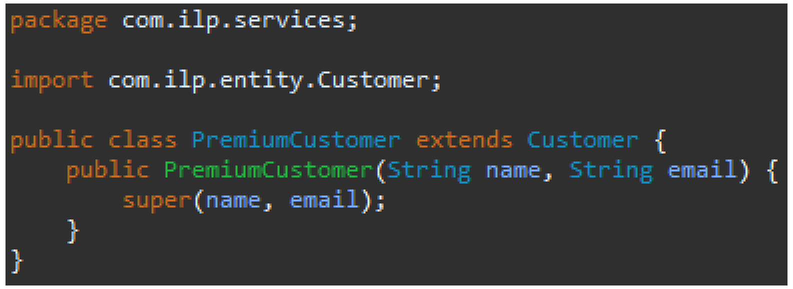


```java
package com.ilp.services;

import com.ilp.entity.Issue;

public class AdvancedIssueResolver implements IssueResolver {
    @Override
    public void resolveIssue(Issue issue) {
        issue.resolveIssue(true);
        System.out.println("A ticket has been raised against your issue");
        System.out.println("Advanced issue resolution applied for: " + issue.getDescription());


        try {
            Thread.sleep(2000); // 5000 milliseconds (5 seconds)
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Your ticket is resolved. Thank you!");
    }
}
```

LISKOV SUBSTITUTION PRINCIPLE (LSP)

PremiumCustomer Class

The **PremiumCustomer** class extends the **Customer** class, honoring the Liskov Substitution Principle by behaving as a subtype of its parent class. This ensures that instances of **PremiumCustomer** can be used wherever instances of **Customer** are expected.

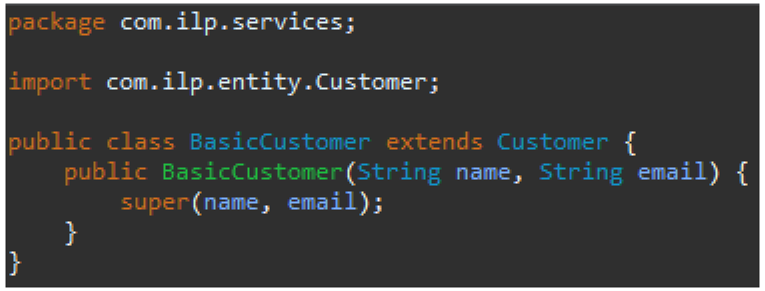


```java
package com.ilp.services;

import com.ilp.entity.Customer;

public class PremiumCustomer extends Customer {
    public PremiumCustomer(String name, String email) {
        super(name, email);
    }
}
```

LISKOV SUBSTITUTION PRINCIPLE (LSP)

BasicCustomer Class

The **BasicCustomer** class extends the **Customer** class, honoring the Liskov Substitution Principle by behaving as a subtype of its parent class. This ensures that instances of **BasicCustomer** can be used wherever instances of **Customer** are expected.



```java
package com.ilp.services;

import com.ilp.entity.Customer;

public class BasicCustomer extends Customer {
    public BasicCustomer(String name, String email) {
        super(name, email);
    }
}
```

INTERFACE SEGREGATION PRINCIPLE (ISP)

CustomerInformation and IssueInformation Interfaces

The ISP is applied by creating specific interfaces (**CustomerInformation** and **IssueInformation**) tailored to their respective responsibilities, preventing clients from being forced to depend on interfaces they do not use.

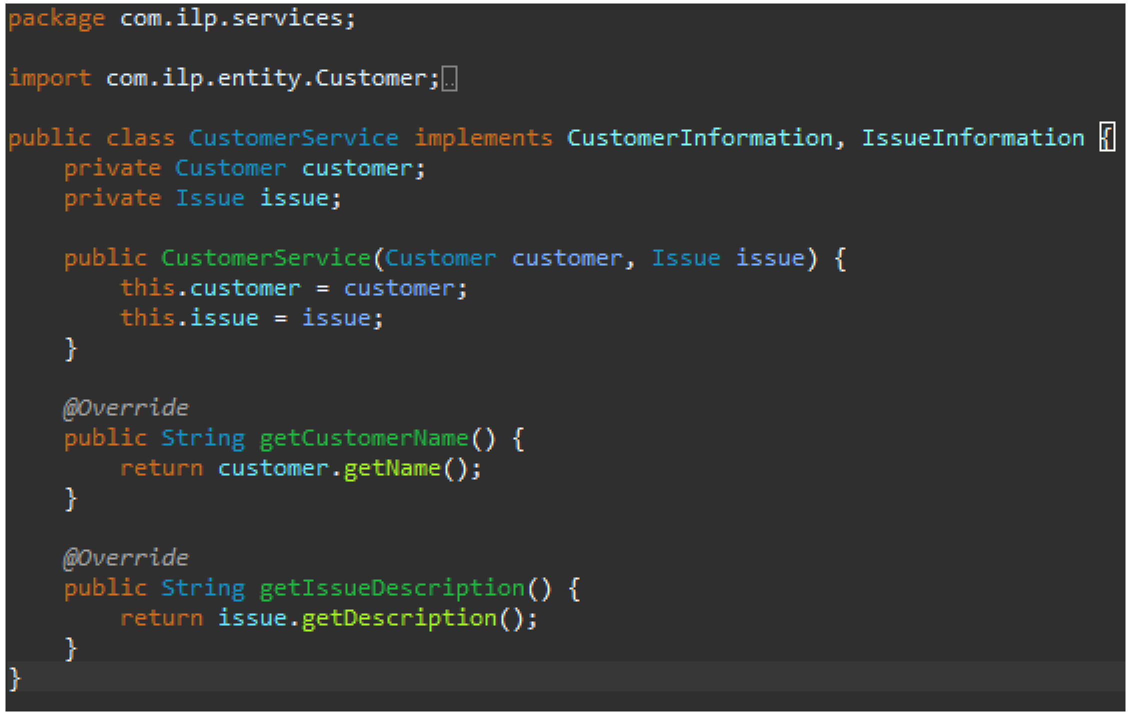
```java
package com.ilp.interfaces;

public interface IssueInformation {
String getIssueDescription();
}
```

```java
package com.ilp.interfaces;

public interface CustomerInformation
{
String getCustomerName();
}
```

INTERFACE SEGREGATION PRINCIPLE (ISP)

CustomerService Class

The **CustomerService** class implements both the **CustomerInformation** and **IssueInformation** interfaces. It acts as a convenient data holder for customer and issue information in specific scenarios.

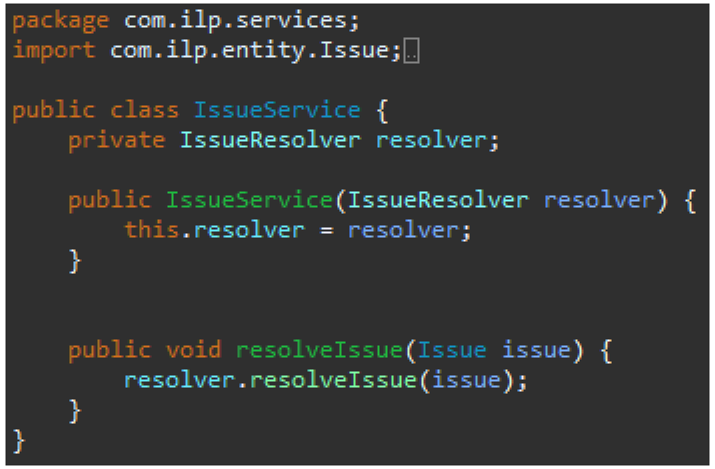


```java
package com.ilp.services;

import com.ilp.entity.Customer;

public class CustomerService implements CustomerInformation, IssueInformation {
    private Customer customer;
    private Issue issue;

    public CustomerService(Customer customer, Issue issue) {
        this.customer = customer;
        this.issue = issue;
    }

    @Override
    public String getCustomerName() {
        return customer.getName();
    }

    @Override
    public String getIssueDescription() {
        return issue.getDescription();
    }
}
```

DEPENDENCY INVERSION PRINCIPLE (DIP)

IssueService Class

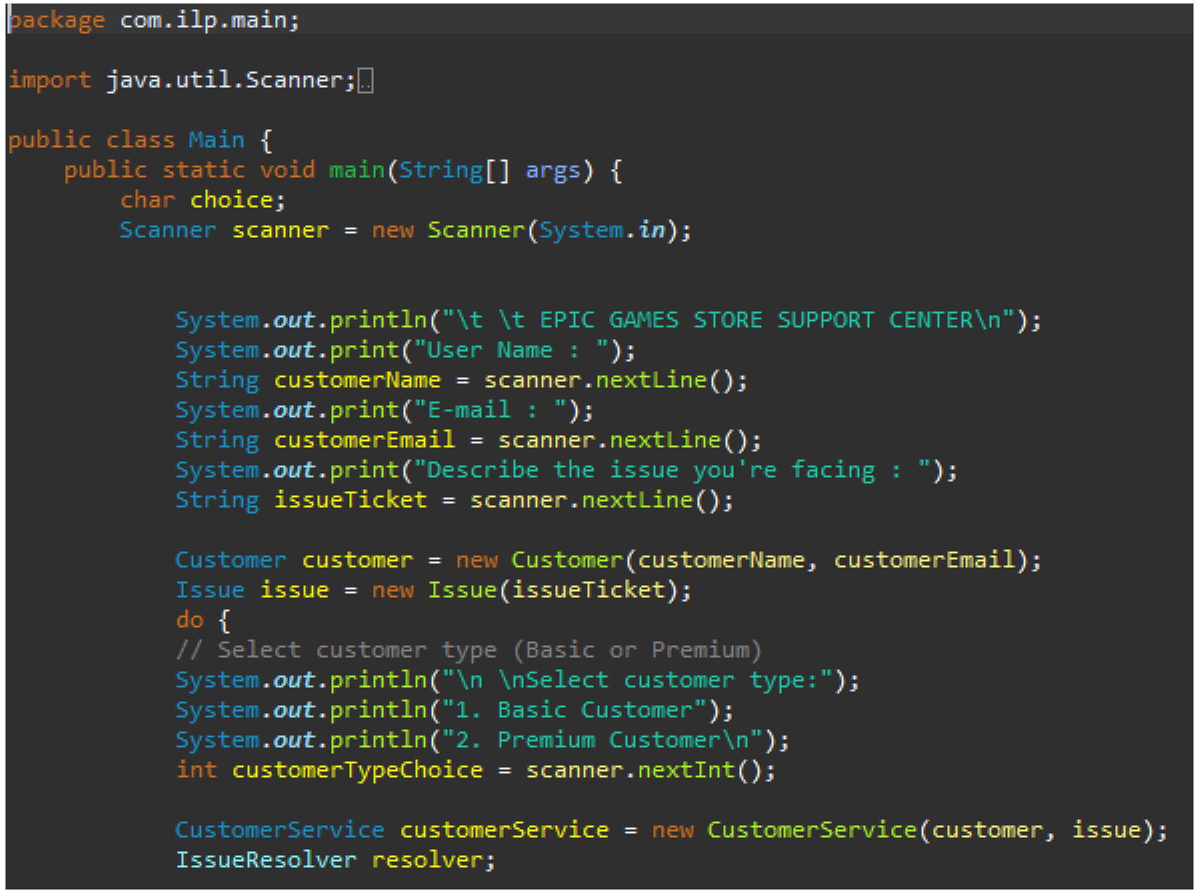
The **IssueService** class adheres to the Dependency Inversion Principle by depending on abstractions (e.g., **IssueResolver** interface) rather than concrete implementations. This promotes flexibility and ease of extension.



```
package com.ilp.services;
import com.ilp.entity.Issue;

public class IssueService {
    private IssueResolver resolver;

    public IssueService(IssueResolver resolver) {
        this.resolver = resolver;
    }


    public void resolveIssue(Issue issue) {
        resolver.resolveIssue(issue);
    }
}
```

USAGE EXAMPLES

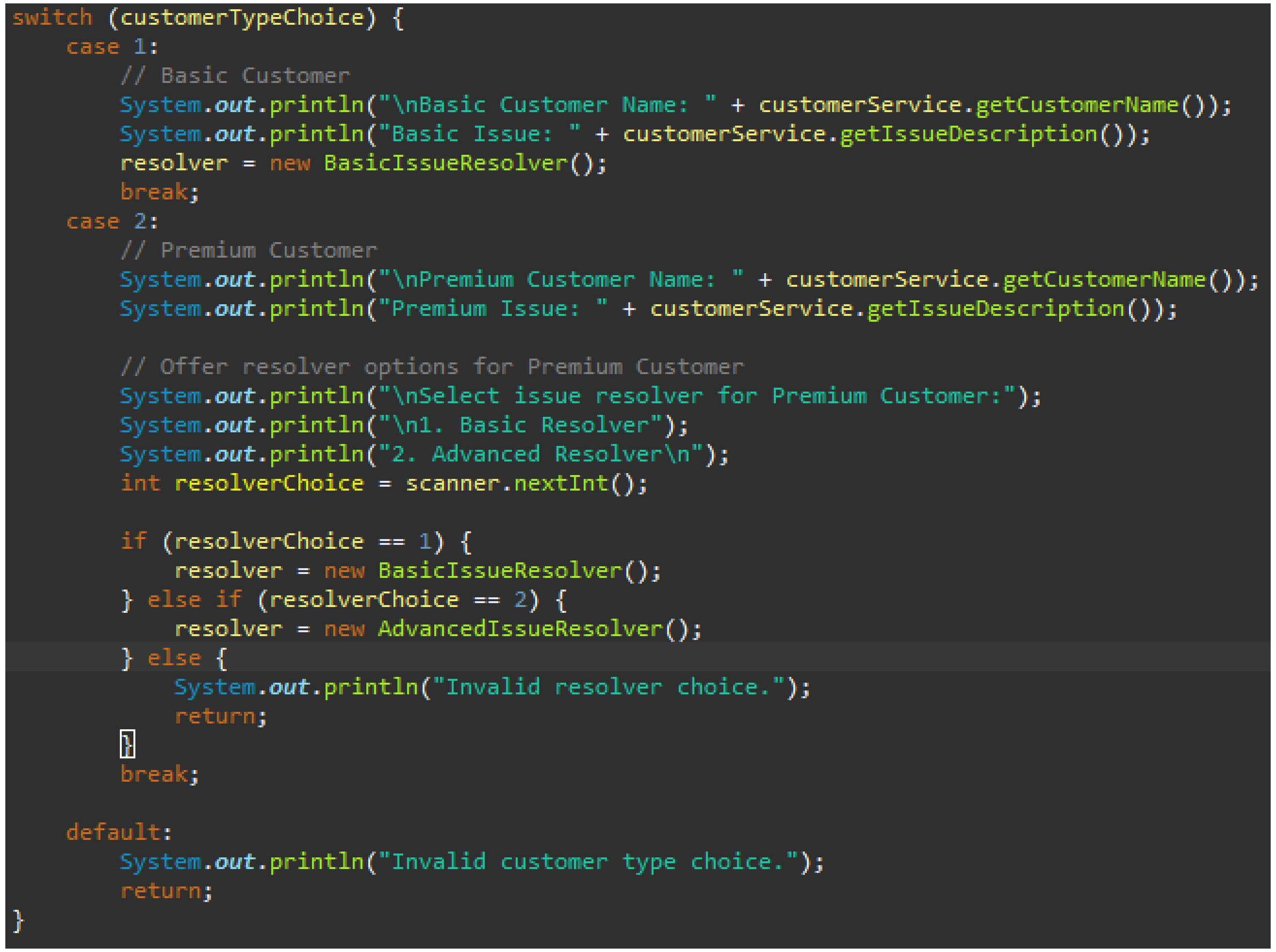
The **Main** class demonstrates the usage of the implemented classes and principles. It showcases the creation of customers, issues, resolution using different strategies, and the introduction of a premium customer with additional functionality.



```
package com.ilp.main;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        char choice;
        Scanner scanner = new Scanner(System.in);


            System.out.println("\t \t EPIC GAMES STORE SUPPORT CENTER\n");
            System.out.print("User Name : ");
            String customerName = scanner.nextLine();
            System.out.print("E-mail : ");
            String customerEmail = scanner.nextLine();
            System.out.print("Describe the issue you're facing : ");
            String issueTicket = scanner.nextLine();

            Customer customer = new Customer(customerName, customerEmail);
            Issue issue = new Issue(issueTicket);
            do {
            // Select customer type (Basic or Premium)
            System.out.println("\n \nSelect customer type:");
            System.out.println("1. Basic Customer");
            System.out.println("2. Premium Customer\n");
            int customerTypeChoice = scanner.nextInt();

            CustomerService customerService = new CustomerService(customer, issue);
            IssueResolver resolver;
```

USAGE EXAMPLES

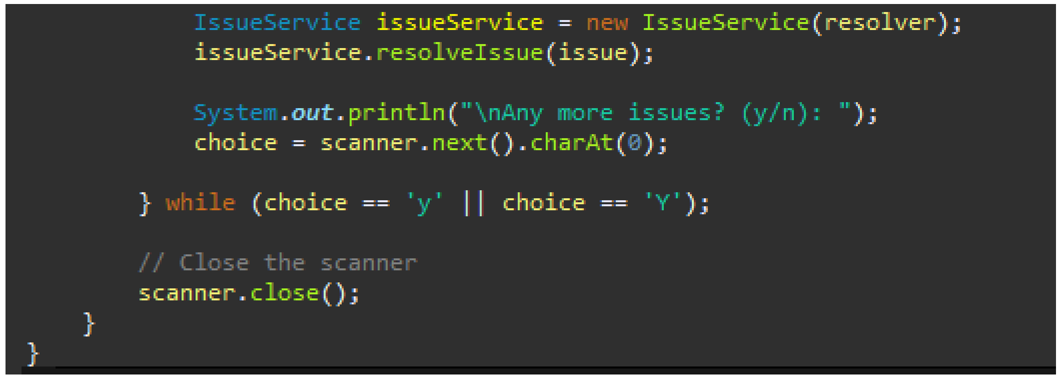
The **Main** class demonstrates the usage of the implemented classes and principles. It showcases the creation of customers, issues, resolution using different strategies, and the introduction of a premium customer with additional functionality.



```java
switch (customerTypeChoice) {
    case 1:
        // Basic Customer
        System.out.println("\nBasic Customer Name: " + customerService.getCustomerName());
        System.out.println("Basic Issue: " + customerService.getIssueDescription());
        resolver = new BasicIssueResolver();
        break;
    case 2:
        // Premium Customer
        System.out.println("\nPremium Customer Name: " + customerService.getCustomerName());
        System.out.println("Premium Issue: " + customerService.getIssueDescription());

        // Offer resolver options for Premium Customer
        System.out.println("\nSelect issue resolver for Premium Customer:");
        System.out.println("\n1. Basic Resolver");
        System.out.println("2. Advanced Resolver\n");
        int resolverChoice = scanner.nextInt();

        if (resolverChoice == 1) {
            resolver = new BasicIssueResolver();
        } else if (resolverChoice == 2) {
            resolver = new AdvancedIssueResolver();
        } else {
            System.out.println("Invalid resolver choice.");
            return;
        }
        break;

    default:
        System.out.println("Invalid customer type choice.");
        return;
}
```

USAGE EXAMPLES

The **Main** class demonstrates the usage of the implemented classes and principles. It showcases the creation of customers, issues, resolution using different strategies, and the introduction of a premium customer with additional functionality.



```
                IssueService issueService = new IssueService(resolver);
                issueService.resolveIssue(issue);

                System.out.println("\nAny more issues? (y/n): ");
                choice = scanner.next().charAt(0);

            } while (choice == 'y' || choice == 'Y');

            // Close the scanner
            scanner.close();
        }
    }
```