Splitwise (LLD Project)

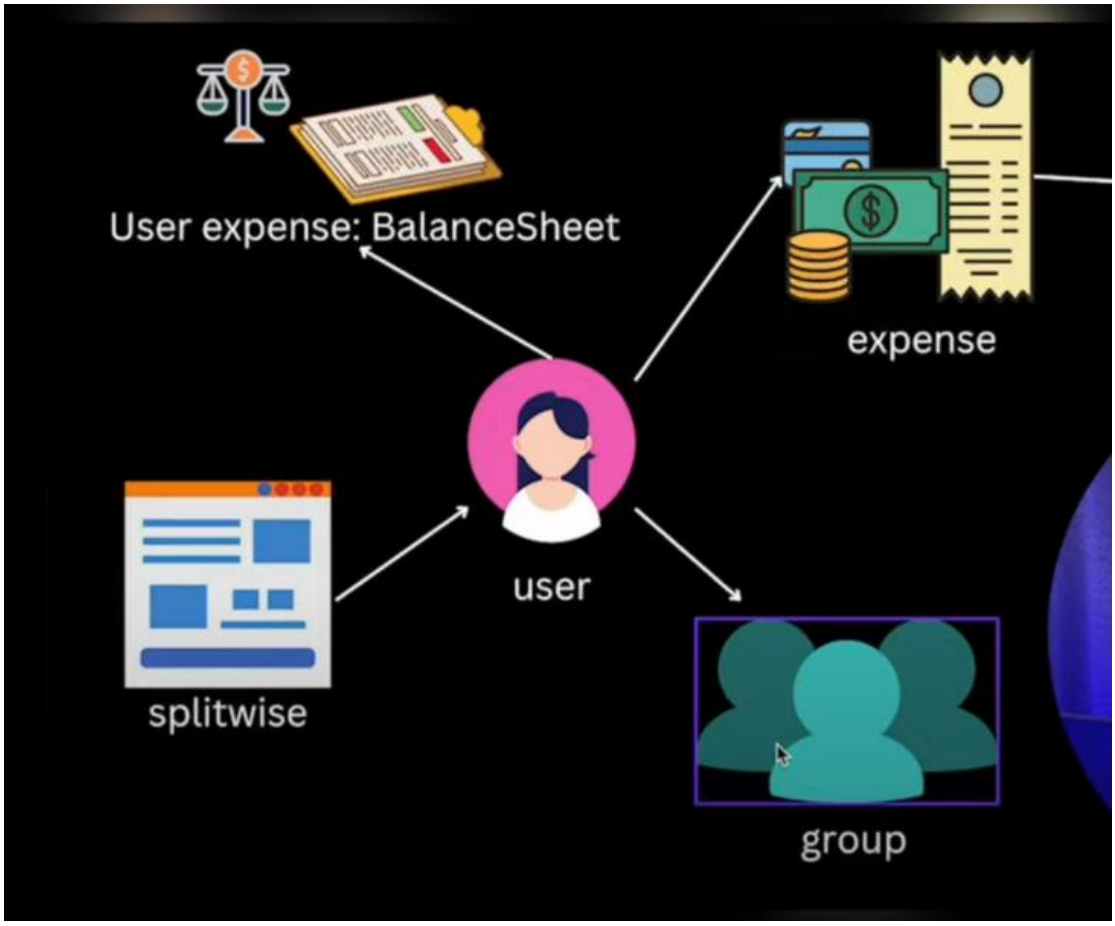
Problem Statement

Design and implement a Splitwise-like system that manages expenses among users, supports different split strategies (equal, unequal, and percentage-based), handles groups, and maintains user balances.

Requirements

1. Add users
2. Create groups of users
3. Add expenses paid by one user and split among others
4. Support for equal, percentage, and unequal split
5. Track balances between users
6. Show balances

Flowchart



Core Entities and Their Code

1. User

Represents an individual in the system.

```
public class User {
    private String userId;
    private String userName;

    public User(String userId, String userName) {
        this.userId = userId;
        this.userName = userName;
    }

    public String getUserId() {
        return userId;
    }

    public String getUserName() {
        return userName;
    }
}
```

---

2. Group

Represents a group of users sharing expenses.

```
public class Group {
    private String groupId;
    private String groupName;
    private List<User> users;

    public Group(String groupId, String groupName, List<User> users) {
        this.groupId = groupId;
        this.groupName = groupName;
        this.users = users;
    }

    public List<User> getUsers() {
        return users;
    }
}
```

---

3. Expense

Stores details of an expense.

```
public class Expense {
    private String expenseId;
    private double amount;
    private User paidBy;
    private List<Split> splits;
    private ExpenseSplitType expenseSplitType;
```

```
    public Expense(String expenseId, double amount, User paidBy, List<Split> splits, ExpenseSplitType
expenseSplitType) {
        this.expenseId = expenseId;
        this.amount = amount;
        this.paidBy = paidBy;
        this.splits = splits;
        this.expenseSplitType = expenseSplitType;
    }
}
```

---

4. Split (Abstract Class)

Represents how the expense is divided among users.

```
public abstract class Split {
    private User user;
    private double amount;

    public Split(User user) {
        this.user = user;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public double getAmount() {
        return amount;
    }
}
```

---

5. Split Strategies

Define how expenses are split.

Equal Split

```
public class EqualExpenseSplit implements ExpenseSplitStrategy {
    public void split(Expense expense) {
        double equalAmount = expense.getAmount() / expense.getSplits().size();
        for (Split split : expense.getSplits()) {
            split.setAmount(equalAmount);
        }
    }
}
```

Percentage Split

```
public class PercentageExpenseSplit implements ExpenseSplitStrategy {
    public void split(Expense expense) {
        for (Split split : expense.getSplits()) {
            double share = expense.getAmount() * split.getAmount() / 100.0;
            split.setAmount(share);
        }
    }
```

```
}
```

Unequal Split

```
public class UnequalExpenseSplit implements ExpenseSplitStrategy {
    public void split(Expense expense) {
        // Presumes that split amounts are already set
    }
}
```

---

6. UserExpenseBalanceSheet

Maintains balances owed between users.

```
public class UserExpenseBalanceSheet {
    private Map<String, Double> userBalanceMap = new HashMap<>();

    public void addBalance(String userId, double amount) {
        userBalanceMap.put(userId, userBalanceMap.getOrDefault(userId, 0.0) + amount);
    }

    public Map<String, Double> getBalances() {
        return userBalanceMap;
    }
}
```

---

7. Splitwise (Main Logic)

Handles overall operations and orchestrates expense addition.

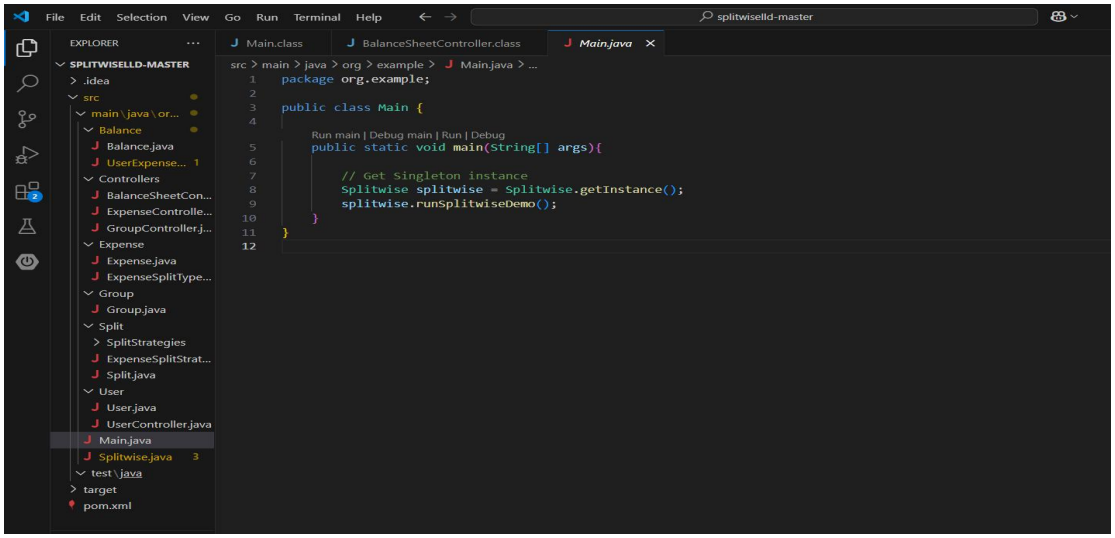
```
public class Splitwise {
    private Map<String, User> userMap;
    private Map<String, UserExpenseBalanceSheet> balanceSheetMap;

    public Splitwise() {
        this.userMap = new HashMap<>();
        this.balanceSheetMap = new HashMap<>();
    }

    public void addUser(User user) {
        userMap.put(user.getUserId(), user);
        balanceSheetMap.put(user.getUserId(), new UserExpenseBalanceSheet());
    }

    public void addExpense(Expense expense) {
        expense.getExpenseSplitType().getStrategy().split(expense);
        User paidBy = expense.getPaidBy();
        for (Split split : expense.getSplits()) {
            if (!split.getUser().getUserId().equals(paidBy.getUserId())) {
                balanceSheetMap.get(split.getUser().getUserId()).addBalance(paidBy.getUserId(),
split.getAmount());
                balanceSheetMap.get(paidBy.getUserId()).addBalance(split.getUser().getUserId(), -
split.getAmount());
            }
        }
```

```
    }

    public void showBalances() {
        for (String userId : balanceSheetMap.keySet()) {
            for (Map.Entry<String, Double> entry : balanceSheetMap.get(userId).getBalances().entrySet())
{
                if (entry.getValue() > 0) {
                    System.out.println(userMap.get(userId).getUserName() + " owes " +
                            userMap.get(entry.getKey()).getUserName() + ": " + entry.getValue());
                }
            }
        }
    }
}
```

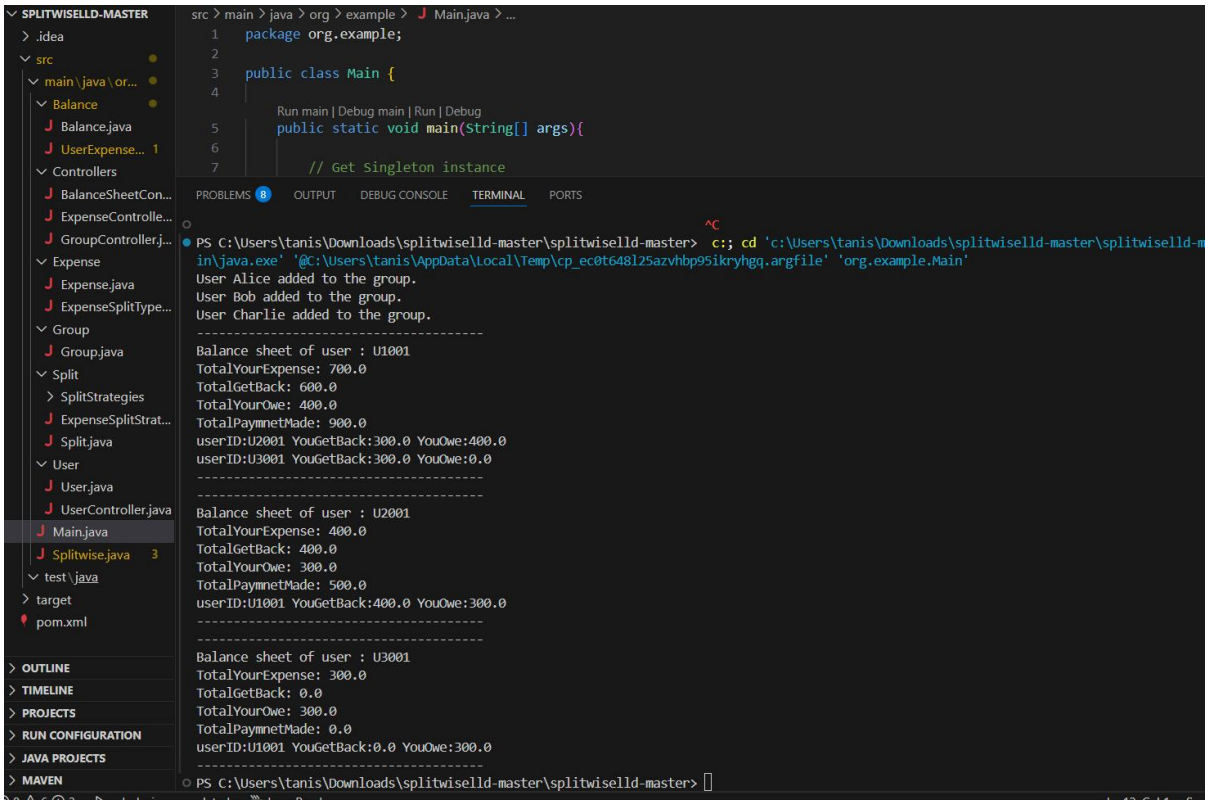
Example Usage (Main.java)

```
public class Main {
    public static void main(String[] args) {
        Splitwise splitwise = new Splitwise();

        User u1 = new User("u1", "Tanish");
        User u2 = new User("u2", "Rahul");

        splitwise.addUser(u1);
        splitwise.addUser(u2);

        List<Split> splits = new ArrayList<>();
        splits.add(new EqualSplit(u1));
        splits.add(new EqualSplit(u2));

        Expense expense = new Expense("e1", 1000, u1, splits, ExpenseSplitType.EQUAL);
        splitwise.addExpense(expense);

        splitwise.showBalances();
    }
}
```

Main.java



Code Output


SPLITWISELLD-MASTER
.idea
src
main\java\or...
Balance
Balance.java
UserExpense... 1
Controllers
BalanceSheetCon...
ExpenseControlle...
GroupController.j...
Expense
Expense.java
ExpenseSplitType...
Group
Group.java
Split
SplitStrategies
ExpenseSplitStrat...
Split.java
User
User.java
UserController.java
Main.java
Splitwise.java 3
test\java
target
pom.xml
OUTLINE
TIMELINE
PROJECTS
RUN CONFIGURATION
JAVA PROJECTS
MAVEN
src > main > java > org > example > Main.java > ...
1 package org.example;
2
3 public class Main {
4
Run main | Debug main | Run | Debug
5 public static void main(String[] args){
6
7 // Get Singleton instance
PROBLEMS 8
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
^C
PS C:\Users\tanis\Downloads\splitwiselld-master\splitwiselld-master> c:; cd 'c:\Users\tanis\Downloads\splitwiselld-master\splitwiselld-m
in\java.exe' '@C:\Users\tanis\AppData\Local\Temp\cp_ec0t648l25azvhbp95ikryhgq.argfile' 'org.example.Main'
User Alice added to the group.
User Bob added to the group.
User Charlie added to the group.

Balance sheet of user : U1001
TotalYourExpense: 700.0
TotalGetBack: 600.0
TotalYourOwe: 400.0
TotalPaymnetMade: 900.0
userID:U2001 YouGetBack:300.0 YouOwe:400.0
userID:U3001 YouGetBack:300.0 YouOwe:0.0

Balance sheet of user : U2001
TotalYourExpense: 400.0
TotalGetBack: 400.0
TotalYourOwe: 300.0
TotalPaymnetMade: 500.0
userID:U1001 YouGetBack:400.0 YouOwe:300.0

Balance sheet of user : U3001
TotalYourExpense: 300.0
TotalGetBack: 0.0
TotalYourOwe: 300.0
TotalPaymnetMade: 0.0
userID:U1001 YouGetBack:0.0 YouOwe:300.0

PS C:\Users\tanis\Downloads\splitwiselld-master\splitwiselld-master>