Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

CS 5800.01 - Advanced Software Engineering

Vending Machine

Code:

```
package Assignment8;

public class CheetosDispenseHandler extends SnackDispenseHandler {

    public CheetosDispenseHandler(SnackDispenseHandler handler){
        super(handler);
    }

    @Override
    public void dispenseSnack(Snack snack, int quantity){
        if(snack.getSnackName().equalsIgnoreCase("cheetos")){
            System.out.printf("This is your %s%n", snack.getSnackName());
            System.out.printf("Thank you for your order. See you soon.\n");
            snack.setSnackQuantity(snack.getSnackQuantity() - quantity);
        }else{
            super.dispenseSnack(snack, quantity);
        }
    }
}
```

```
package Assignment8;

public class CokeDispenseHandler extends SnackDispenseHandler {

    public CokeDispenseHandler(SnackDispenseHandler handler){
        super(handler);
    }

    @Override
    public void dispenseSnack(Snack snack, int quantity){
        if(snack.getSnackName().equalsIgnoreCase("coke")){
            System.out.printf("This is your %s%n", snack.getSnackName());
            System.out.printf("Thank you for your order. See you soon.\n");
            snack.setSnackQuantity(snack.getSnackQuantity() - quantity);
        }else{
            super.dispenseSnack(snack, quantity);
        }
    }
}
```

```
package Assignment8;

public class DispensingState implements StateOfVendingMachine{
```

```
    private VendingMachine vendingMachine;
    private SnackDispenseHandler snackDispenseHandler;

    public DispensingState(VendingMachine vendingMachine) {
        this.vendingMachine = vendingMachine;
        snackDispenseHandler = new CokeDispenseHandler(
                new PepsiDispenseHandler(new CheetosDispenseHandler(
                        new DoritosDispenseHandler(new
KitKatDispenseHandler(new SnickersDispenseHandler(null))))));
    }

    @Override
    public void selectSnack(String snackName, int quantity) {
        System.out.println("You can't pick another snack until you receive
your previous snack.");
    }

    @Override
    public void insertCoin(double userPaidAmount) {
        System.out.println("You can't insert another coin until you receive
your snack.");
    }

    @Override
    public void dispenseSnack() {
        Snack selectedSnack = vendingMachine.getSelectedSnack();
        int quantity = vendingMachine.getSelectedSnackQuantity();
        if (selectedSnack.getSnackQuantity() >= quantity) {
            snackDispenseHandler.dispenseSnack(selectedSnack, quantity);
            vendingMachine.setState(new IdleState(vendingMachine));
        } else {
            System.out.println("Insufficient stock to fulfill your order.
Please try a different snack or quantity.");
            vendingMachine.setState(new IdleState(vendingMachine));
        }
    }


}
```

```
package Assignment8;

public class DoritosDispenseHandler extends SnackDispenseHandler {
    public DoritosDispenseHandler(SnackDispenseHandler handler){
        super(handler);
    }

    @Override
    public void dispenseSnack(Snack snack, int quantity){
        if(snack.getSnackName().equalsIgnoreCase("doritos") &&
snack.getSnackQuantity() > quantity){
            System.out.printf("This is your %s%n", snack.getSnackName());
            System.out.printf("Thank you for your order. Please come
```

```
again.\n");
            snack.setSnackQuantity(snack.getSnackQuantity() - quantity);
        }else{
            super.dispenseSnack(snack, quantity);
        }
    }
}
```

```
package Assignment8;

import java.util.HashMap;
import java.util.Map;

public class Driver {
    public static void main(String[] args) {
        Map<String, Snack> snacks = createSnacks();
        VendingMachine vendingMachine = new VendingMachine();
        vendingMachine.setSnacks(snacks);

        // Case 1: Successful transaction with change returned to customer
        executeScenario(vendingMachine, "Pepsi", 3, 10, "ENOUGH MONEY AND
QUANTITY");

        // Case 2: Not enough money inserted inside the vending machine
        executeScenario(vendingMachine, "Doritos", 4, 2.55, "NOT ENOUGH
MONEY");

        // Case 3: Successful transaction with exact amount
        executeScenario(vendingMachine, "Snickers", 4, 10, "QUANTITY HITS 0
WITH SNICKERS");

        // Case 3.1: Attempting to buy more than available in stock.
        executeScenario(vendingMachine, "Snickers", 5, 15, "NO MORE QUANTITY
OF SNACK");
    }

    private static void executeScenario(VendingMachine vendingMachine, String
snackName, int quantity, double coin, String caseDescription) {
        System.out.println("\n--------------------" + caseDescription + "----
----------------");
        vendingMachine.selectSnack(snackName, quantity);
        vendingMachine.insertCoin(coin);
        vendingMachine.dispenseSnack();
    }

    public static Map<String, Snack> createSnacks() {
        Map<String, Snack> snacks = new HashMap<>();
        //Coke, Pepsi, Cheetos, Doritos, KitKat, and Snickers
        snacks.put("Coke", new Snack("Coke", 1.39, 3));
        snacks.put("Pepsi", new Snack("Pepsi", 1.49, 4));
        snacks.put("Cheetos", new Snack("Cheetos", 2.55, 5));
        snacks.put("Doritos", new Snack("Doritos", 3.60, 6));
        snacks.put("KitKat", new Snack("KitKat", 1.89, 7));
```

```
        snacks.put("Snickers", new Snack("Snickers", 1.39, 8));

        System.out.println("Welcome to the Vending Machine!");
        System.out.println("Available Snacks:");
        for (Map.Entry<String, Snack> entry : snacks.entrySet()) {
            Snack snack = entry.getValue();
            System.out.printf("%-10s | Quantity: %-5d | Price: $%-5.2f
each\n", snack.getSnackName(), snack.getSnackQuantity(),
snack.getSnackPrice());
        }
        System.out.println();
        return snacks;
    }
}
```

```
package Assignment8;

public class IdleState implements StateOfVendingMachine{

    private VendingMachine vendingMachine;

    public IdleState(VendingMachine vendingMachine) {
        this.vendingMachine = vendingMachine;
    }

    public void showSnackOrderedInfo(String snackName, int quantity){
        System.out.printf("%d %s is selected - ", quantity, snackName);
        System.out.printf("%s price: $%.2f/each\n", snackName,
vendingMachine.getSnacks().get(snackName).getSnackPrice());
    }

    public boolean hasSnackAvailable() {
        if (vendingMachine.getSelectedSnack().getSnackQuantity() > 0) {
            return true;
        } else {
            return false;
        }
    }

    @Override
    public void selectSnack(String snackName, int quantity) {
        showSnackOrderedInfo(snackName, quantity);
        vendingMachine.setState(new WaitingForMoneyState(vendingMachine));
    }

    @Override
    public void insertCoin(double userPaidAmount) {
        System.out.printf("You haven't picked a snack yet. Please pick a
snack first.\n");
    }

    @Override
    public void dispenseSnack() {
        System.out.println("You haven't picked a snack yet. Please pick a
```

```
snack first.");
    }
}
```

```
package Assignment8;

public class KitKatDispenseHandler extends SnackDispenseHandler {

    public KitKatDispenseHandler(SnackDispenseHandler handler){
        super(handler);
    }

    @Override
    public void dispenseSnack(Snack snack, int quantity){
        if(snack.getSnackName().equalsIgnoreCase("kitkat")){
            System.out.printf("This is your %s%n", snack.getSnackName());
            System.out.printf("Thank you for your order. See you soon.\n");
            snack.setSnackQuantity(snack.getSnackQuantity() - quantity);
        }else{
            super.dispenseSnack(snack, quantity);
        }
    }
}
```

```
package Assignment8;

public class PepsiDispenseHandler extends SnackDispenseHandler {
    public PepsiDispenseHandler(SnackDispenseHandler handler){
        super(handler);
    }

    @Override
    public void dispenseSnack(Snack snack, int quantity){
        if(snack.getSnackName().equalsIgnoreCase("pepsi")){
            System.out.printf("This is your %s%n", snack.getSnackName());
            System.out.printf("Thank you for your order. See you soon.\n");
            snack.setSnackQuantity(snack.getSnackQuantity() - quantity);
        }else{
            super.dispenseSnack(snack, quantity);
        }
    }
}
```

```
package Assignment8;

public class Snack {
    private String snackName;
    private double snackPrice;
    private int snackQuantity;
```

```
    public Snack(String snackName, double snackPrice, int snackQuantity) {
        this.snackName = snackName;
        this.snackPrice = snackPrice;
        this.snackQuantity = snackQuantity;
    }

    public int getSnackQuantity() {
        return snackQuantity;
    }

    public double getSnackPrice() {
        return snackPrice;
    }

    public String getSnackName() {
        return snackName;
    }

    public void setSnackQuantity(int snackQuantity) {
        this.snackQuantity = snackQuantity;
    }

    public int getQuantity() {
        return snackQuantity;
    }
}
```

```
package Assignment8;

public abstract class SnackDispenseHandler {
    private SnackDispenseHandler next;

    public SnackDispenseHandler(SnackDispenseHandler next){
        this.next = next;
    }

    public void dispenseSnack(Snack snack, int quantity){
        if(next != null){
            next.dispenseSnack(snack, quantity);
        } else if(next == null || snack.getSnackQuantity() < quantity){
            System.out.printf("Snack %s is not available\n",
snack.getSnackName());
        }
    }

}
```

```
package Assignment8;


public class SnickersDispenseHandler extends SnackDispenseHandler {
```

```
    public SnickersDispenseHandler(SnackDispenseHandler handler){
        super(handler);
    }

    @Override
    public void dispenseSnack(Snack snack, int quantity){
        if(snack.getSnackName().equalsIgnoreCase("snickers")){
            System.out.printf("This is your %s%n", snack.getSnackName());
            System.out.printf("Thank you for your order. See you soon.\n");
            snack.setSnackQuantity(snack.getSnackQuantity() - quantity);
        }else{
            super.dispenseSnack(snack, quantity);
        }
    }
}
```

```
package Assignment8;

interface StateOfVendingMachine {
    public void selectSnack(String snackName, int quantity);
    public void insertCoin(double userPaidAmount);
    public void dispenseSnack();
}
```

```
package Assignment8;

import java.util.HashMap;
import java.util.Map;

public class VendingMachine {

    private StateOfVendingMachine state;
    private Map<String, Snack> snacks;
    private Snack userSelectedSnack;
    private int userSelectedSnackQuantity;

    public VendingMachine() {
        state = new IdleState(this);
        snacks = new HashMap<String, Snack>();
    }

    public void setSnacks(Map<String, Snack> snacks) {
        this.snacks = snacks;
    }

    public void setState(StateOfVendingMachine state) {
        this.state = state;
    }

    public void selectSnack(String snackName, int quantity){
        if (!snacks.containsKey(snackName)) {
            System.out.println("Snack not available in this vending machine.
```

```
Try some other vending machine nearby");
            return;
        }
        userSelectedSnack = snacks.get(snackName);
        userSelectedSnackQuantity = quantity;
        state.selectSnack(snackName, quantity);
    }

    public void insertCoin(double coinAmount) {
        state.insertCoin(coinAmount);
    }

    public void dispenseSnack() {
        state.dispenseSnack();
    }

    public Map<String,Snack> getSnacks() {
        return snacks;
    }

    public Snack getSelectedSnack() {
        return userSelectedSnack;
    }

    public int getSelectedSnackQuantity() {
        return userSelectedSnackQuantity;
    }

    public void showCurrentStateInfor(){
        System.out.printf("Current state: %s\n",
state.getClass().getSimpleName());
    }

}
```

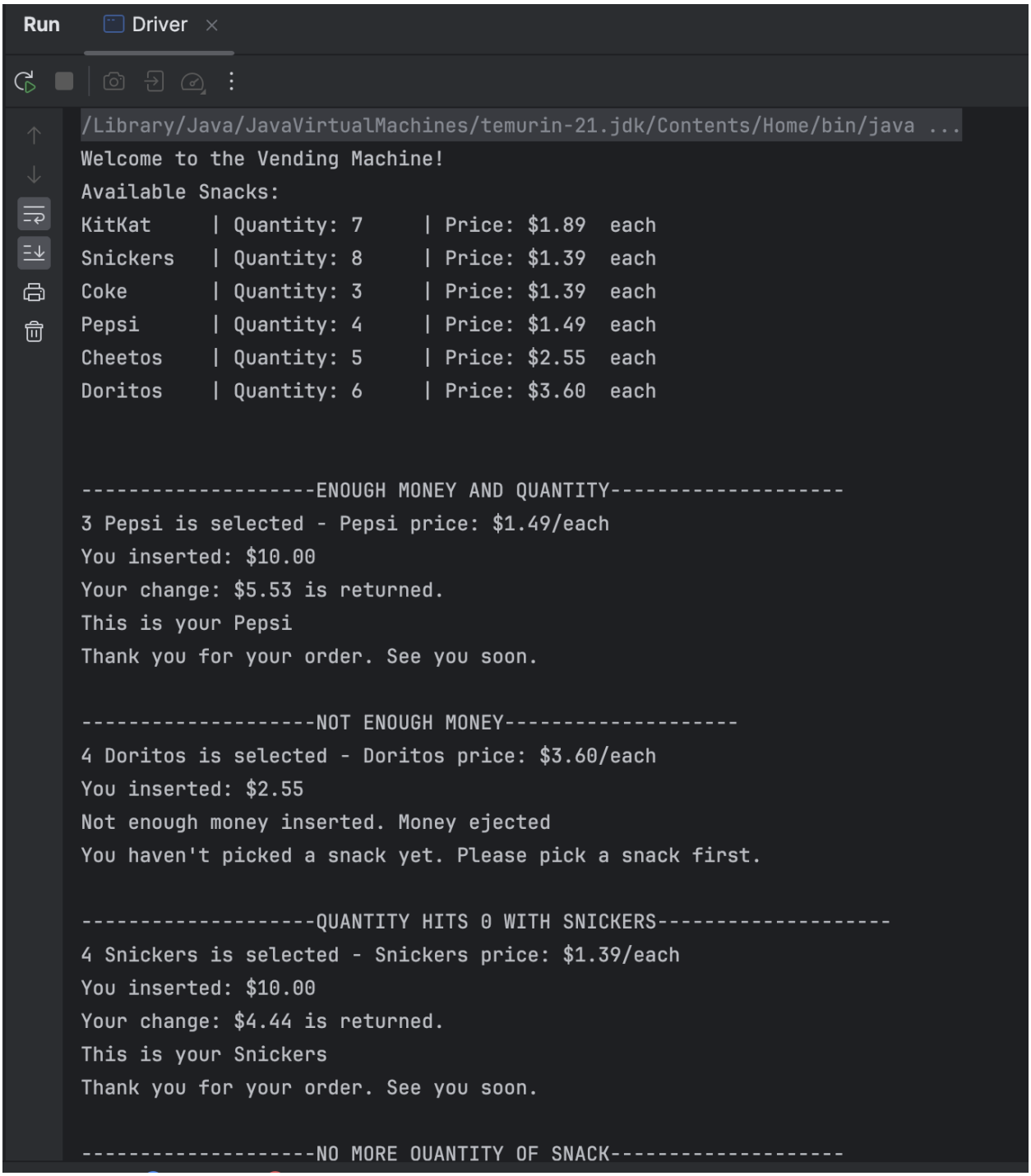
```
package Assignment8;

public class WaitingForMoneyState implements StateOfVendingMachine{
    private VendingMachine vendingMachine;

    public WaitingForMoneyState(VendingMachine vendingMachine) {
        this.vendingMachine = vendingMachine;
    }


    @Override
    public void selectSnack(String snackName, int quantity) {
        System.out.printf("You can't choose another snack until you pay for
your current snack.\n");
    }

    @Override
    public void insertCoin(double userPaidAmount) {
        double snackPrice =
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

```
vendingMachine.getSelectedSnack().getSnackPrice();
        double userSnackQuantity = vendingMachine.getSelectedSnackQuantity();
        double actualSnackQuantity =
vendingMachine.getSelectedSnack().getSnackQuantity();
        double totalCost = snackPrice * userSnackQuantity;
        System.out.printf("You inserted: $%.2f\n", userPaidAmount);
        if (userPaidAmount < totalCost){
            System.out.println("Not enough money inserted. Money ejected");
            vendingMachine.setState(new IdleState(vendingMachine));
            return;
        } else if(actualSnackQuantity < userSnackQuantity) {
            System.out.printf("Not enough %s in the machine now. Money
ejected\n", vendingMachine.getSelectedSnack().getSnackName());
            vendingMachine.setState(new IdleState(vendingMachine));
            return;
        } else if(userPaidAmount > totalCost){
            double returnAmount = userPaidAmount - totalCost;
            System.out.printf("Your change: $%.2f is returned.\n",
returnAmount);
        }
        vendingMachine.setState(new DispensingState(vendingMachine));
    }

    @Override
    public void dispenseSnack() {
        System.out.println("Customers must pay first");
    }
}
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

Output:



Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

```
--------------------NO MORE QUANTITY OF SNACK--------------------
5 Snickers is selected - Snickers price: $1.39/each
You inserted: $15.00
Not enough Snickers in the machine now. Money ejected
You haven't picked a snack yet. Please pick a snack first.

Process finished with exit code 0
```

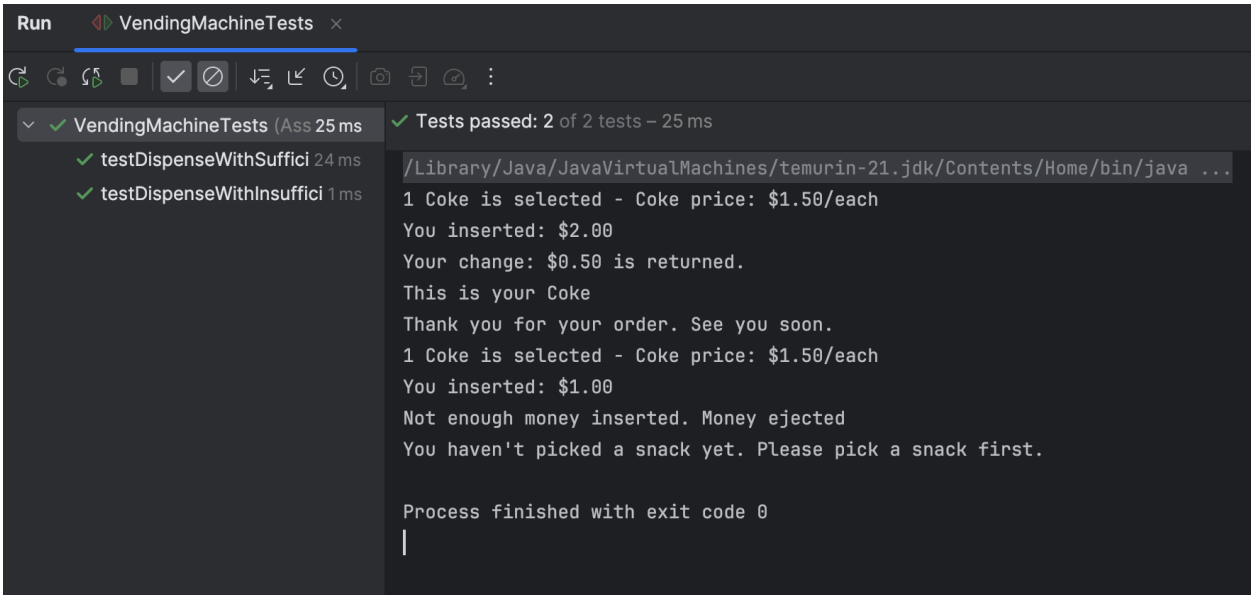
Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

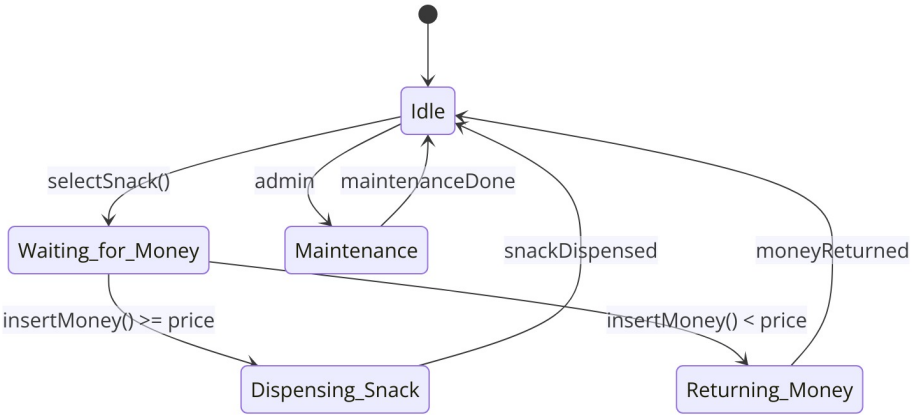
Test code:

```
package Assignment8;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import java.util.Map;

public class VendingMachineTests {

    private VendingMachine vendingMachine;

    @BeforeEach
    public void setUp() {
        vendingMachine = new VendingMachine();
        Map<String, Snack> snacks = Map.of(
                "Coke", new Snack("Coke", 1.50, 5)
        );
        vendingMachine.setSnacks(snacks);
    }

    @Test
    public void testDispenseWithSufficientFunds() {
        vendingMachine.selectSnack("Coke", 1);
        vendingMachine.insertCoin(2.00);
        vendingMachine.dispenseSnack();
        Snack coke = vendingMachine.getSnacks().get("Coke");
        assertEquals(4, coke.getSnackQuantity(), "Coke quantity should be
reduced by one.");
    }

    @Test
    public void testDispenseWithInsufficientFunds() {
        vendingMachine.selectSnack("Coke", 1);
        vendingMachine.insertCoin(1.00); // Not enough for a Coke priced at
$1.50
        vendingMachine.dispenseSnack();
        Snack coke = vendingMachine.getSnacks().get("Coke");
        assertEquals(5, coke.getSnackQuantity(), "Coke quantity should not
change.");
    }
}
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

Test output:





Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

UML

Part 1

Code:

```
@startuml

class Character {
  - character: char
  - characterProperty: CharacterProperties
  + Character(character: char, characterProperty: CharacterProperties)
  + setCharacter(character: char): void
  + setCharacterProperty(characterProperty: CharacterProperties): void
  + getCharacter(): char
  + getFont(): String
  + getColor(): String
  + getSize(): int
}

class CharacterProperties {
  - font: String
  - color: String
  - size: int
  + CharacterProperties(font: String, color: String, size: int)
  + getFont(): String
  + getColor(): String
  + getSize(): int
}

class CharacterPropertiesFactory {
  - flyweights: Map<String, CharacterProperties>
  + setAndRetrieveFlyweightCharacterProperties(font: String, color: String,
size: int): CharacterProperties
  + sizeOfMap(): int
}

class Document {
  - characters: List<Character>
  - propertiesFactory: CharacterPropertiesFactory
  + addCharacter(c: char, font: String, color: String, size: int): void
  + editCharacterProperties(index: int, font: String, color: String, size:
int): void
  + saveToFile(filename: String): void
  + loadFromFile(filename: String): void
}

class Driver {
  + main(args: String[]): void
}

Character "1" *-- "1" CharacterProperties
Document "1" *-- "1" CharacterPropertiesFactory
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

```
Document "1" *-- "*" Character
Driver -- Document

@enduml
```

Output:

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

Part 2

Code:

```
@startuml
'https://plantuml.com/object-diagram
together {

    object "<u>index0: Character</u>" as Character0{
        character="H"
        characterProperty=
        Arial16Black
    }
    object "<u>index1: Character</u>" as Character1{
        character="e"
        characterProperty=
        Arial14Black
    }
    object "<u>index2: Character</u>" as Character2{
        character="l"
        characterProperty=
        Arial14Black
    }
    object "<u>index3: Character</u>" as Character3{
        character="l"
        characterProperty=
        Arial14Black
    }
    object "<u>index4: Character</u>" as Character4{
        character="o"
        characterProperty=
        Arial14Black
    }
    object "<u>index5: Character</u>" as Character5{
        character="W"
        characterProperty=
        Arial16Black
    }
    object "<u>index6: Character</u>" as Character6{
        character="o"
        characterProperty=
        Arial14Black
    }
    object "<u>index7: Character</u>" as Character7{
        character="r"
        characterProperty=
        Arial14Black
    }
    object "<u>index8: Character</u>" as Character8{
        character="l"
        characterProperty=
        Arial14Black
    }
    object "<u>index9: Character</u>" as Character9{
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

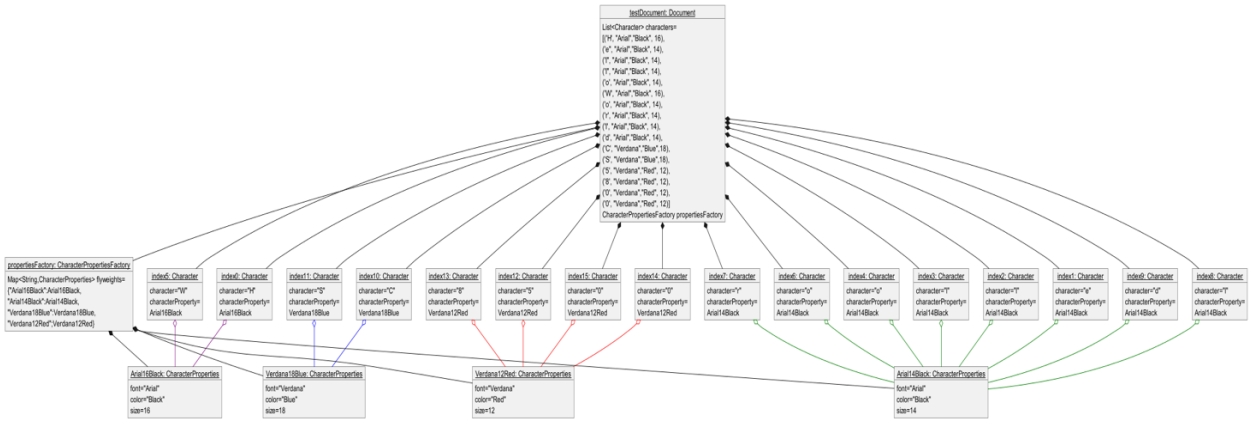
```
        character="d"
        characterProperty=
        Arial14Black
    }
    object "<u>index10: Character</u>" as Character10{
        character="C"
        characterProperty=
        Verdana18Blue
    }
    object "<u>index11: Character</u>" as Character11{
        character="S"
        characterProperty=
        Verdana18Blue
    }
    object "<u>index12: Character</u>" as Character12{
        character="5"
        characterProperty=
        Verdana12Red
    }
    object "<u>index13: Character</u>" as Character13{
        character="8"
        characterProperty=
        Verdana12Red
    }
    object "<u>index14: Character</u>" as Character14{
        character="0"
        characterProperty=
        Verdana12Red
    }
    object "<u>index15: Character</u>" as Character15{
        character="0"
        characterProperty=
        Verdana12Red
    }
}
object "<u>Arial16Black: CharacterProperties</u>" as Arial16Black{
    font="Arial"
    color="Black"
    size=16
}
object "<u>Arial14Black: CharacterProperties</u>" as Arial14Black{
    font="Arial"
    color="Black"
    size=14
}
object "<u>Verdana18Blue: CharacterProperties</u>" as Verdana18Blue{
    font="Verdana"
    color="Blue"
    size=18
}
object "<u>Verdana12Red: CharacterProperties</u>" as Verdana12Red{
    font="Verdana"
    color="Red"
    size=12
}
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

```
Character0 o--[#purple] Arial16Black
Character1 o--[#green] Arial14Black
Character2 o--[#green] Arial14Black
Character3 o--[#green] Arial14Black
Character4 o--[#green] Arial14Black
Character5 o--[#purple] Arial16Black
Character6 o--[#green] Arial14Black
Character7 o--[#green] Arial14Black
Character8 o--[#green] Arial14Black
Character9 o--[#green] Arial14Black
Character10 o--[#blue] Verdana18Blue
Character11 o--[#blue] Verdana18Blue
Character12 o--[#red] Verdana12Red
Character13 o--[#red] Verdana12Red
Character14 o--[#red] Verdana12Red
Character15 o--[#red] Verdana12Red

object "<u>testDocument: Document</u>" as Document{
  List<Character> characters=
  [('H', "Arial","Black", 16),
  ('e', "Arial","Black", 14),
  ('l', "Arial","Black", 14),
  ('l', "Arial","Black", 14),
  ('o', "Arial","Black", 14),
  ('W', "Arial","Black", 16),
  ('o', "Arial","Black", 14),
  ('r', "Arial","Black", 14),
  ('l', "Arial","Black", 14),
  ('d', "Arial","Black", 14),
  ('C', "Verdana","Blue",18),
  ('S', "Verdana","Blue",18),
  ('5', "Verdana","Red", 12),
  ('8', "Verdana","Red", 12),
  ('0', "Verdana","Red", 12),
  ('0', "Verdana","Red", 12)]

  CharacterPropertiesFactory propertiesFactory
}
object "<u>propertiesFactory: CharacterPropertiesFactory</u>" as
flyweightFactory{
    Map<String,CharacterProperties> flyweights=
    {"Arial16Black":Arial16Black,
     "Arial14Black":Arial14Black,
     "Verdana18Blue":Verdana18Blue,
     "Verdana12Red";Verdana12Red}
}

Document *--flyweightFactory

flyweightFactory *--Arial16Black
flyweightFactory *--Arial14Black
flyweightFactory *--Verdana18Blue
flyweightFactory *--Verdana12Red

Document *-- Character0
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

```
Document *-- Character1
Document *-- Character2
Document *-- Character3
Document *-- Character4
Document *-- Character5
Document *-- Character6
Document *-- Character7
Document *-- Character8
Document *-- Character9
Document *-- Character10
Document *-- Character11
Document *-- Character12
Document *-- Character13
Document *-- Character14
Document *-- Character15

@enduml
```



Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

Part 3

Code:

```
@startuml
'https://plantuml.com/object-diagram
together{
    object "<u>index0: Character</u>" as Character0{
        character="H"
        characterProperty=
        Calibri99Blue
    }
    object "<u>index1: Character</u>" as Character1{
        character="e"
        characterProperty=
        Arial14Black
    }
    object "<u>index2: Character</u>" as Character2{
        character="l"
        characterProperty=
        Arial14Black
    }
    object "<u>index3: Character</u>" as Character3{
        character="l"
        characterProperty=
        Arial14Black
    }
    object "<u>index4: Character</u>" as Character4{
        character="o"
        characterProperty=
        Arial14Black
    }
    object "<u>index5: Character</u>" as Character5{
        character="W"
        characterProperty=
        Calibri99Blue
    }
    object "<u>index6: Character</u>" as Character6{
        character="o"
        characterProperty=
        Arial14Black
    }
    object "<u>index7: Character</u>" as Character7{
        character="r"
        characterProperty=
        Arial14Black
    }
    object "<u>index8: Character</u>" as Character8{
        character="l"
        characterProperty=
        Arial14Black
```

```
    }
    object "<u>index9: Character</u>" as Character9{
        character="d"
        characterProperty=
        Arial14Black
    }
    object "<u>index10: Character</u>" as Character10{
        character="C"
        characterProperty=
        Verdana18Blue
    }
    object "<u>index11: Character</u>" as Character11{
        character="S"
        characterProperty=
        Verdana18Blue
    }
    object "<u>index12: Character</u>" as Character12{
        character="5"
        characterProperty=
        Verdana12Red
    }
    object "<u>index13: Character</u>" as Character13{
        character="8"
        characterProperty=
        Verdana12Red
    }
    object "<u>index14: Character</u>" as Character14{
        character="0"
        characterProperty=
        Verdana12Red
    }
    object "<u>index15: Character</u>" as Character15{
        character="0"
        characterProperty=
        Verdana12Red
    }
}

object "<u>Arial16Black: CharacterProperties</u>" as Arial16Black{
    font="Arial"
    color="Black"
    size=16
}
object "<u>Arial14Black: CharacterProperties</u>" as Arial14Black{
    font="Arial"
    color="Black"
    size=14
}
object "<u>Verdana18Blue: CharacterProperties</u>" as Verdana18Blue{
    font="Verdana"
    color="Blue"
    size=18
}
object "<u>Verdana12Red: CharacterProperties</u>" as Verdana12Red{
    font="Verdana"
    color="Red"
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

```
    size=12
}
object "<u>Calibri99Blue: CharacterProperties</u>" as Calibri99Blue{
    font="Calibri"
    color="Blue"
    size=99
}


Character0 o--[#purple] Calibri99Blue
Character1 o--[#green] Arial14Black
Character2 o--[#green] Arial14Black
Character3 o--[#green] Arial14Black
Character4 o--[#green] Arial14Black
Character5 o--[#purple] Calibri99Blue
Character6 o--[#green] Arial14Black
Character7 o--[#green] Arial14Black
Character8 o--[#green] Arial14Black
Character9 o--[#green] Arial14Black
Character10 o--[#blue] Verdana18Blue
Character11 o--[#blue] Verdana18Blue
Character12 o--[#red] Verdana12Red
Character13 o--[#red] Verdana12Red
Character14 o--[#red] Verdana12Red
Character15 o--[#red] Verdana12Red

object "<u>testDocument: Document</u>" as Document{
  List<Character> characters=
  [('H', "Calibri","Blue", 99),
  ('e', "Arial","Black", 14),
  ('l', "Arial","Black", 14),
  ('l', "Arial","Black", 14),
  ('o', "Arial","Black", 14),
  ('W', "Calibri","Blue", 99),
  ('o', "Arial","Black", 14),
  ('r', "Arial","Black", 14),
  ('l', "Arial","Black", 14),
  ('d', "Arial","Black", 14),
  ('C', "Verdana","Blue",18),
  ('S', "Verdana","Blue",18),
  ('5', "Verdana","Red", 12),
  ('8', "Verdana","Red", 12),
  ('0', "Verdana","Red", 12),
  ('0', "Verdana","Red", 12)]

  CharacterPropertiesFactory propertiesFactory
}
object "<u>propertiesFactory: CharacterPropertiesFactory</u>" as
flyweightFactory{
    Map<String,CharacterProperties> flyweights=
    {"Arial16Black":Arial16Black,
     "Arial14Black":Arial14Black,
     "Verdana18Blue":Verdana18Blue,
     "Verdana12Red";Verdana12Red
     "Calibri99Blue":Calibri99Blue}
}
```

Chaitanya Ganpat Nalage
017300803
Github Link: https://github.com/chaitanyanalage/CS5800

```
Document *--flyweightFactory

flyweightFactory *--Arial16Black
flyweightFactory *--Arial14Black
flyweightFactory *--Verdana18Blue
flyweightFactory *--Verdana12Red
flyweightFactory *--Calibri99Blue


Document *-- Character0
Document *-- Character1
Document *-- Character2
Document *-- Character3
Document *-- Character4
Document *-- Character5
Document *-- Character6
Document *-- Character7
Document *-- Character8
Document *-- Character9
Document *-- Character10
Document *-- Character11
Document *-- Character12
Document *-- Character13
Document *-- Character14
Document *-- Character15

@enduml
```

Output: