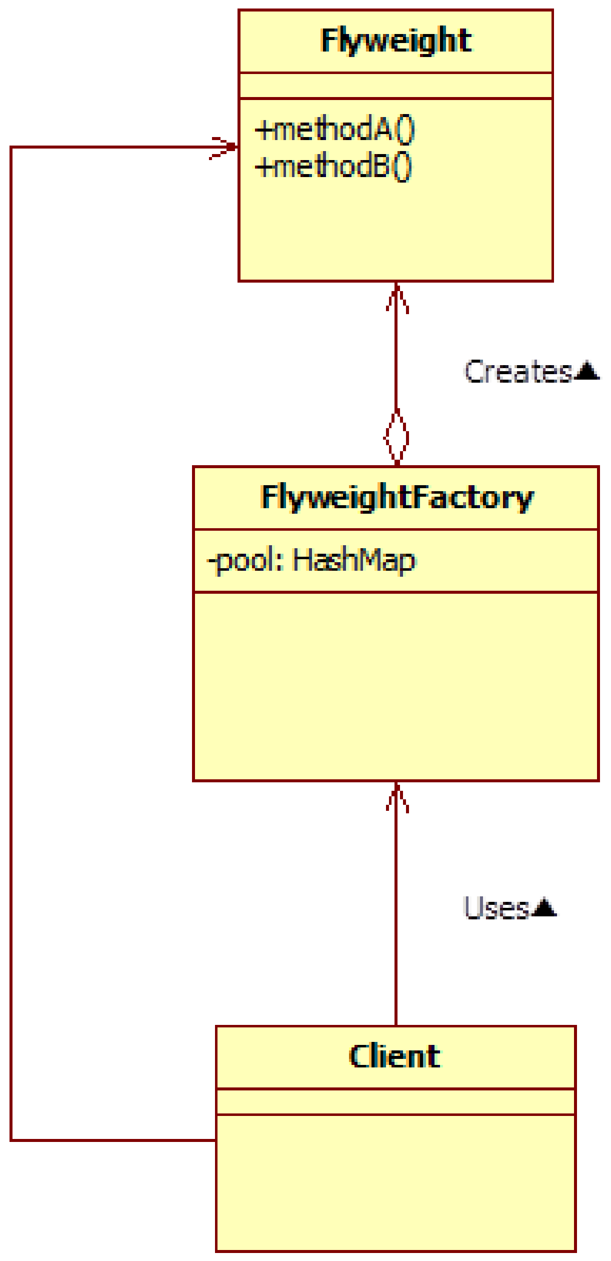
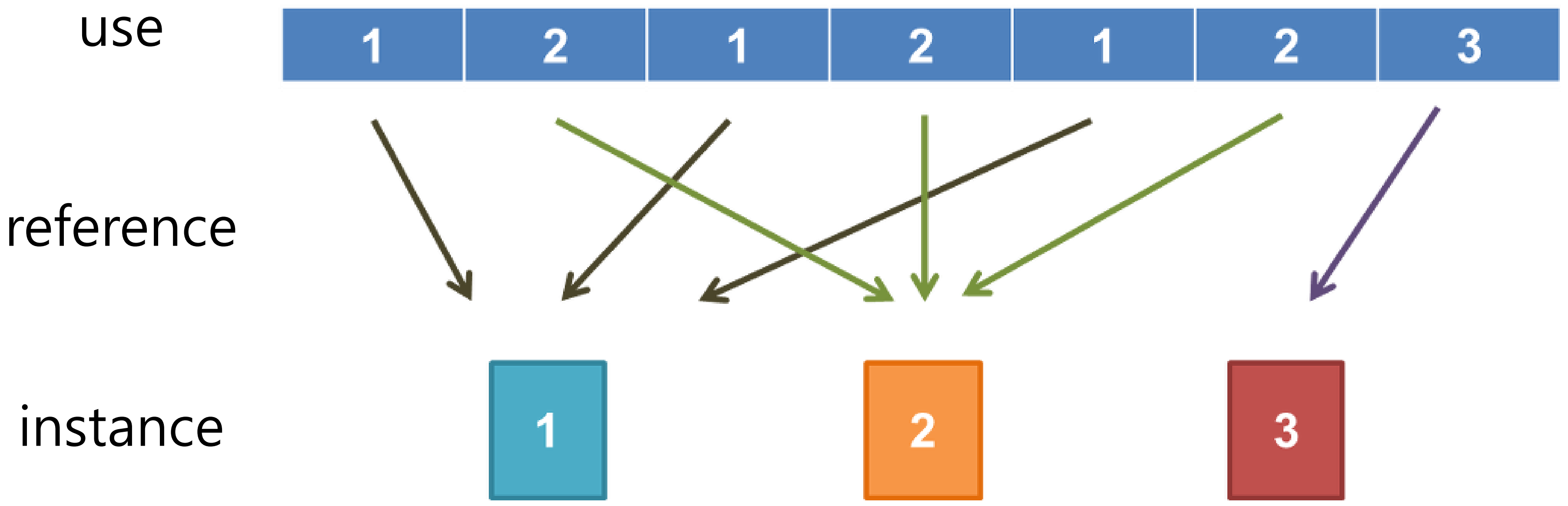
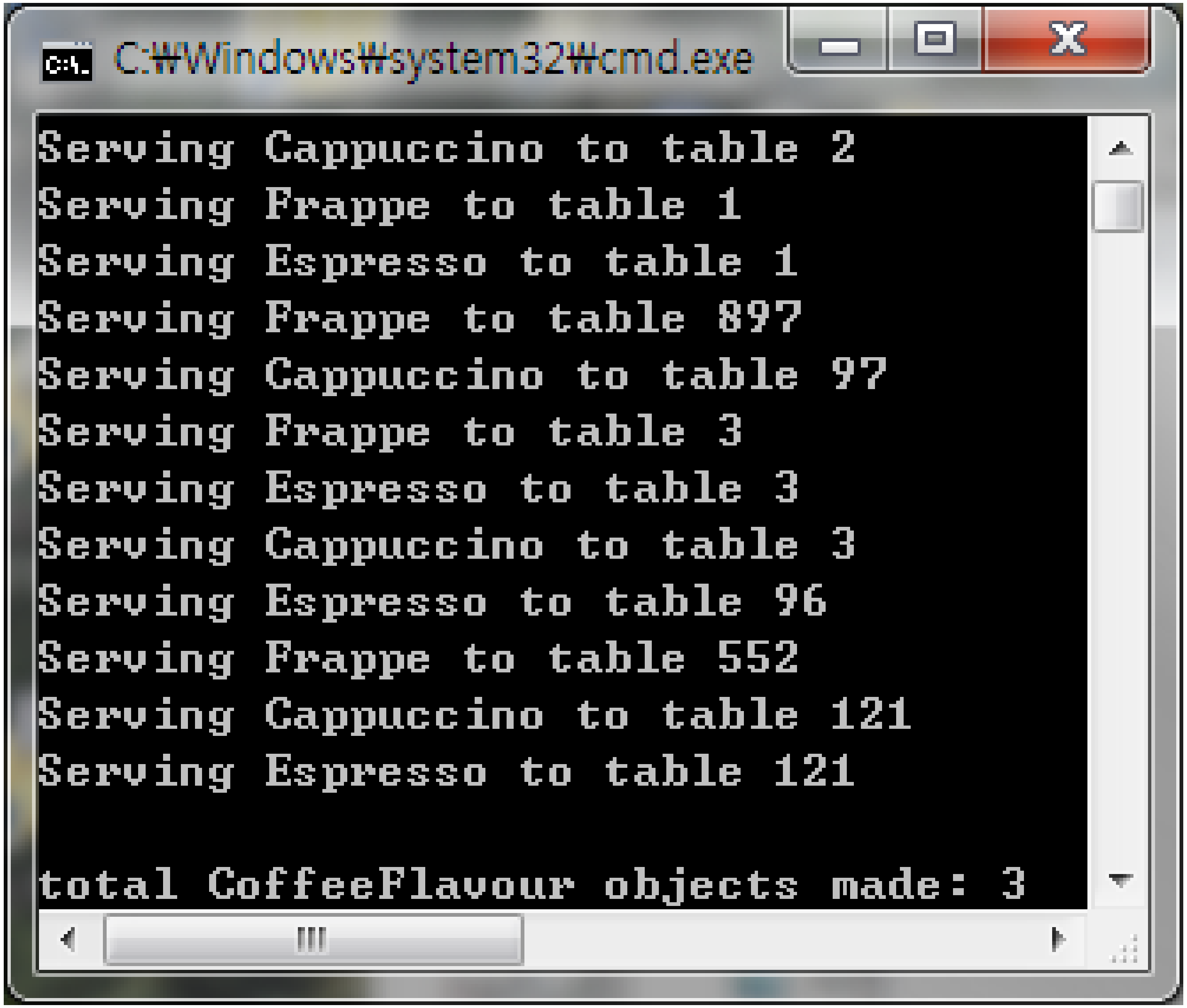
Flyweight Pattern

OODP 2022

Flyweight Pattern
Efficient Use of Sharable Resource




use
reference
instance
1
2
1
2
1
2
3
1
2
3



```
Serving Cappuccino to table 2
Serving Frappe to table 1
Serving Espresso to table 1
Serving Frappe to table 897
Serving Cappuccino to table 97
Serving Frappe to table 3
Serving Espresso to table 3
Serving Cappuccino to table 3
Serving Espresso to table 96
Serving Frappe to table 552
Serving Cappuccino to table 121
Serving Espresso to table 121


total CoffeeFlavour objects made: 3
```

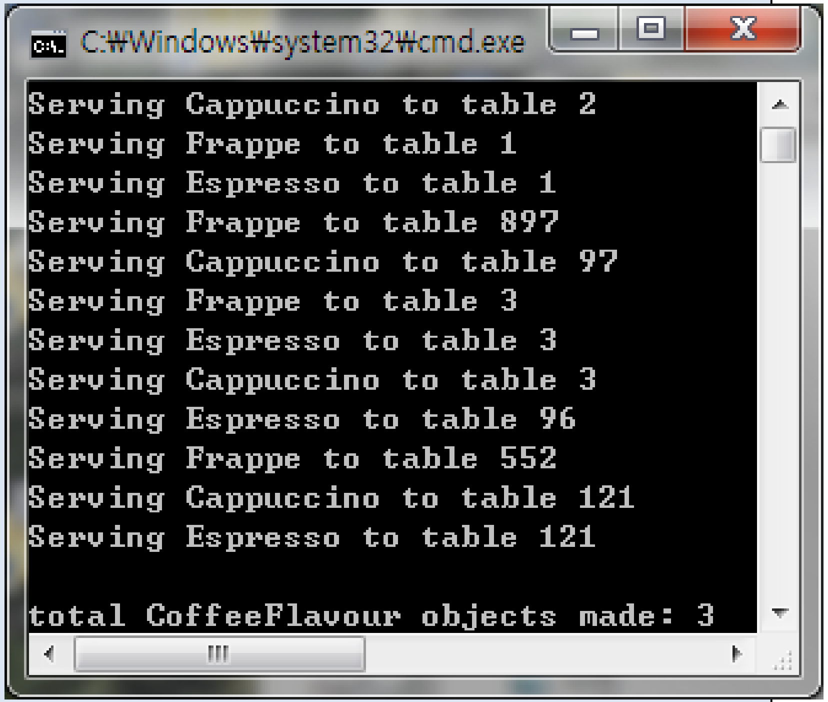
```
class CoffeeFlavour {
  private final String name;
  CoffeeFlavour(String newFlavor) {
    this.name = newFlavor;
  }
 public String toString() {
    return name;
  }
}// Menu acts as a factory and cache for CoffeeFlavour flyweight objects
class Menu {
 private Map<String, CoffeeFlavour> flavours = new HashMap<String,
CoffeeFlavour>();
  CoffeeFlavour lookup(String flavorName) {
    if (!flavours.containsKey(flavorName))
      flavours.put(flavorName, new CoffeeFlavour(flavorName));
    return flavours.get(flavorName);
  }
  int totalCoffeeFlavoursMade() {
    return flavours.size();
  }  }
```

```
class Order {
  private final int tableNumber;
  private final CoffeeFlavour flavour;

  Order(int tableNumber, CoffeeFlavour flavor) {
    this.tableNumber = tableNumber;
    this.flavour = flavor;
  }

  void serve() {
    System.out.println("Serving " + flavour + " to table “
                            + tableNumber);
  }
}
```

```
class CoffeeShop {
  private final List<Order> orders =
          new ArrayList<Order>();
  private final Menu menu = new Menu();
  void takeOrder(String flavourName, int table) {
    CoffeeFlavour flavour = menu.lookup(flavourName);
    Order order = new Order(table, flavour);
    orders.add(order);
  }
  void service() {
    for (Order order : orders)
      order.serve();
  }
  String report() {
    return "₩ntotal CoffeeFlavour objects made: "
         + menu.totalCoffeeFlavoursMade();
  }
```

Question: the difference between flavor and orders

```java
public static void main(String[] args) {
    CoffeeShop shop = new CoffeeShop();
    shop.takeOrder("Cappuccino", 2);
    shop.takeOrder("Frappe", 1);
    shop.takeOrder("Espresso", 1);
    shop.takeOrder("Frappe", 897);
    shop.takeOrder("Cappuccino", 97);
    shop.takeOrder("Frappe", 3);
    shop.takeOrder("Espresso", 3);
    shop.takeOrder("Cappuccino", 3);
    shop.takeOrder("Espresso", 96);
    shop.takeOrder("Frappe", 552);
    shop.takeOrder("Cappuccino", 121);
    shop.takeOrder("Espresso", 121);
    shop.service();
    System.out.println(shop.report());
  }
}
```



```
Serving Cappuccino to table 2
Serving Frappe to table 1
Serving Espresso to table 1
Serving Frappe to table 897
Serving Cappuccino to table 97
Serving Frappe to table 3
Serving Espresso to table 3
Serving Cappuccino to table 3
Serving Espresso to table 96
Serving Frappe to table 552
Serving Cappuccino to table 121
Serving Espresso to table 121

total CoffeeFlavour objects made: 3
```

```
......##........
..######........
......##........
......##........
......##........
......##........
..##########....
................
....######......
..##......##....
..........##....
......####......
....##..........
..##............
..##########....
................
......##........
..######........
......##........
......##........
......##........
......##........
..##########....
................
....######......
..##......##....
..........##....
......####......
....##..........
..##............
..##########....
................
```

```
......##........
..######........
......##........
......##........
......##........
......##........
..##########....
................
....######......
..##......##....
..........##....
......####......
....##..........
..##............
..##########....
................
....######......
..##......##....
..........##....
......####......
..........##....
..##......##....
....######......
................
```

Yuki Book Example

```
public class Main {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Usage: java Main digits");
            System.out.println("Example: java Main 1212123");
            System.exit(0);
        }
        BigString bs = new BigString(args[0]);
        bs.print();
    }
}
```

```
public class BigString {
     private BigChar[] bigchars;
     public BigString(String string) {
        bigchars = new BigChar[string.length()];
       BigCharFactory factory = BigCharFactory.getInstance();
        for (int i = 0; i < bigchars.length; i++) {
            bigchars[i] = factory.getBigChar(string.charAt(i));
        }
    }
    public void print() {
        for (int i = 0; i < bigchars.length; i++) {
            bigchars[i].print();
        }
    }
}
```

```
import java.util.HashMap;
public class BigCharFactory {
    private HashMap pool = new HashMap();
    private static BigCharFactory singleton =
                new BigCharFactory();
    private BigCharFactory() {
    }
    public static BigCharFactory getInstance() {
        return singleton;
    }
    public synchronized BigChar getBigChar(char charname) {
        BigChar bc = (BigChar)pool.get("" + charname);
        if (bc == null) {
            bc = new BigChar(charname);
            pool.put("" + charname, bc);
        }
        return bc;
    }
}
```

question: what is the role of bigchars and pool? their difference?

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class BigChar {
    private char charname;
    private String fontdata;
    public BigChar(char charname) {
        this.charname = charname;
        try {
            BufferedReader reader =
                        new BufferedReader(
                new FileReader("big" + charname + ".txt")
            );
            String line;
            StringBuffer buf = new StringBuffer();
            while ((line = reader.readLine()) != null) {
                buf.append(line);
                buf.append("₩n");
            }
            reader.close();
            this.fontdata = buf.toString();
        } catch (IOException e) {
            this.fontdata = charname + "?";
        }
    }
   public void print() {
        System.out.print(fontdata);
    }
}
```