Monday January 7

Lecture 1

EECS4312

$\boxed{\text{error}}$

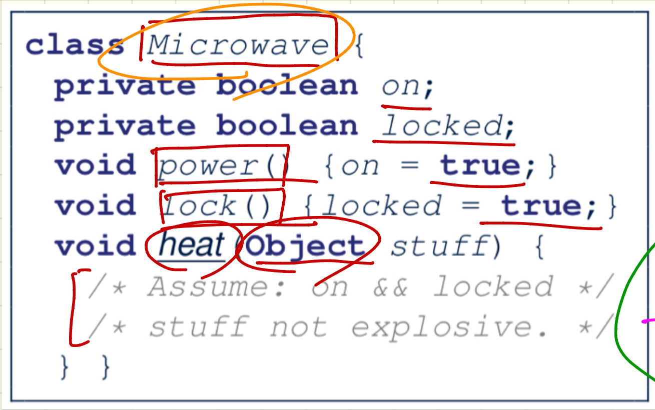
① → Requirement (Informal → computable)

② → Design (– architecture – specification ?)

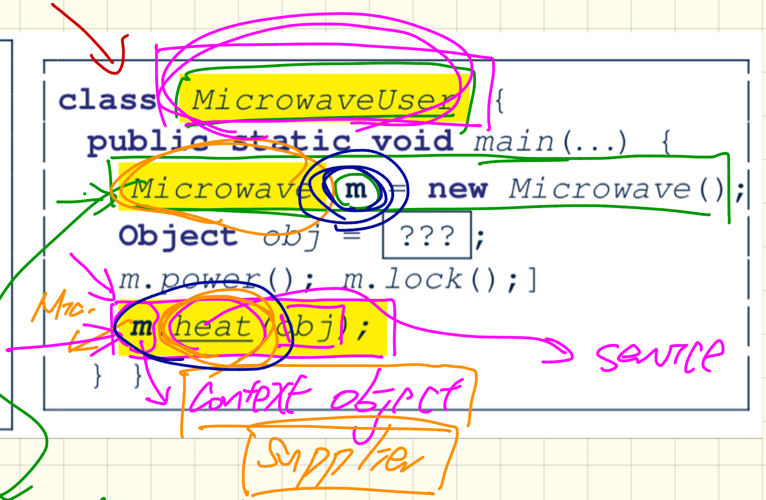
③ → Implementation (efficiency).

④ → Deployment (Windows Mac Mobile).

Client vs. Supplier in OOP

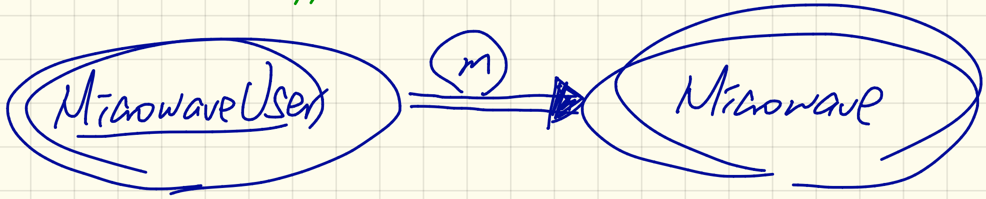


```
class Microwave {
  private boolean on;
  private boolean locked;
  void power() {on = true;}
  void lock() {locked = true;}
  void heat(Object stuff) {
   /* Assume: on && locked */
   /* stuff not explosive. */
  } }
```



```
class MicrowaveUser {
  public static void main(...) {
    Microwave m = new Microwave();
    Object obj = ???;
    m.power(); m.lock();]
    m.heat(obj);
  } }
```

client-supplier relation.



```
class Microwave {
  private boolean on;
  private boolean locked;
  void power() {on = true;}
  void lock() {locked = true;}
  void heat(Object stuff) {
    /* Assume: on && locked */
    /* stuff not explosive. */
} }
```

client

```
class MicrowaveUser {
  public static void main(...) {
    Microwave m = new Microwave();
    Object obj = ???;
    [m.power(); m.lock();]
    m.heat(obj);
} }
```

↳ check on obj.

Q1: Has the client followed the instructions?

We don't know ∵ obj ???

as part of the API of this method; not clear about what will be achieved.

Q2: Has the supplier fulfilled their obligations?

A Simple Design Problems : Bank Accounts

REQ1 : Each account is associated with the *name* of its owner (e.g., "`Jim`") and an integer *balance* that is always positive.

REQ2 : We may *withdraw* an integer amount from an account.

Bank Accounts in Java : Version I

```java
public class AccountV1 {
        private String owner;
        private int balance;
        public String getOwner() { return owner; }
        public int getBalance() { return balance; }
        public AccountV1(String owner, int balance) {
                this.owner = owner; this.balance = balance;
        }
        public void withdraw(int amount) {
                this.balance = this.balance - amount;
        }
        public String toString() {
                return owner + "'s current balance is: " + balance;
        }
}
```

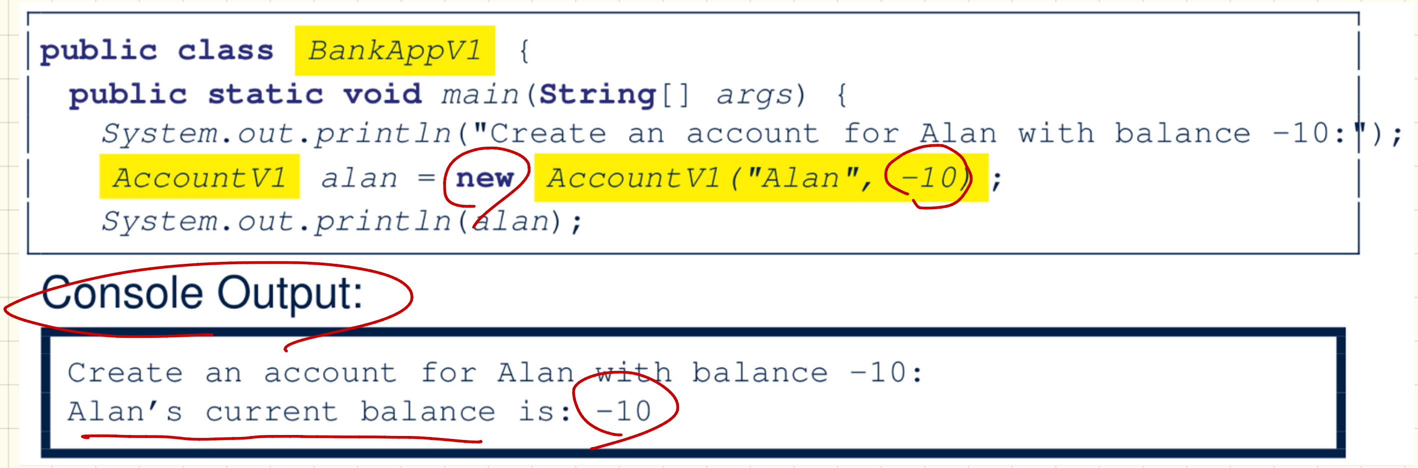
$\rightarrow \emptyset$

-10^6

10^6

-10^6

Bank Accounts in Java: Version 1 Critique (1)



```java
public class BankAppV1 {
  public static void main(String[] args) {
    System.out.println("Create an account for Alan with balance -10:");
    AccountV1 alan = new AccountV1("Alan", -10);
    System.out.println(alan);
```

Console Output:

```
Create an account for Alan with balance -10:
Alan's current balance is: -10
```

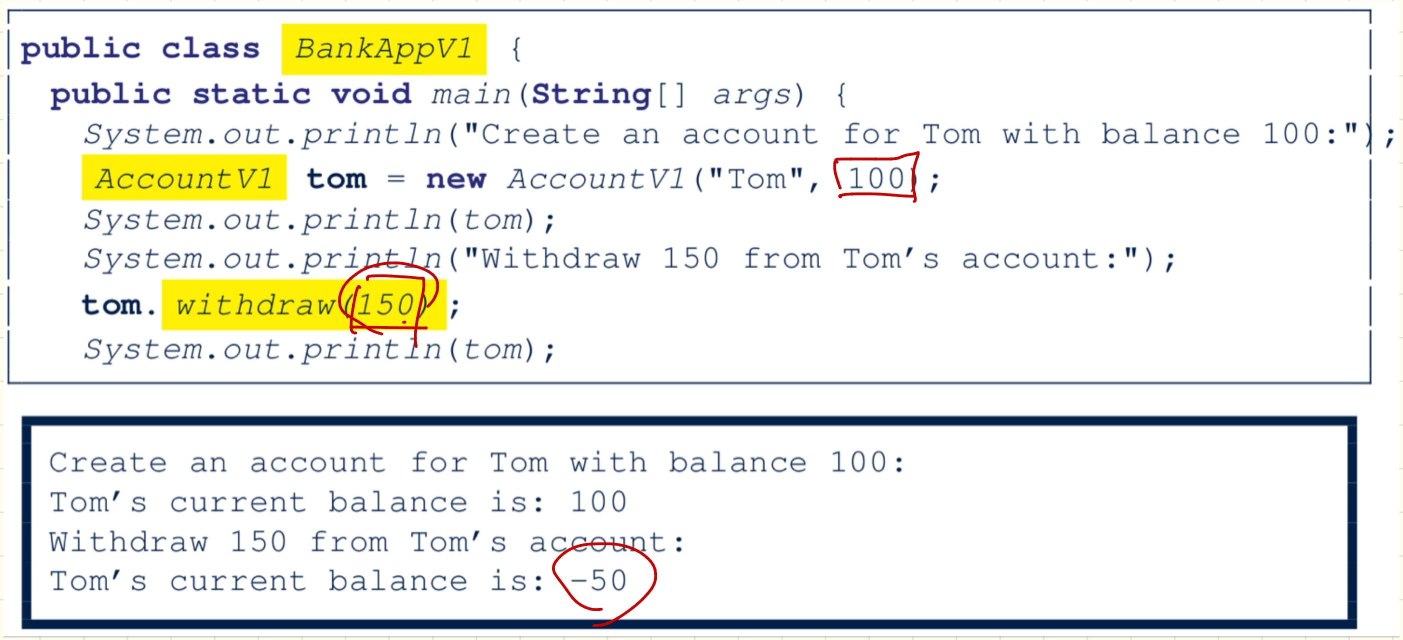
Bank Accounts in Java : Version I Critique (2)

```
public class BankAppV1 {
  public static void main(String[] args) {
    System.out.println("Create an account for Mark with balance 100:");
    AccountV1 mark = new AccountV1("Mark", 100);
    System.out.println(mark);
    System.out.println("Withdraw -1000000 from Mark's account:");
    mark. withdraw(-1000000) ;
    System.out.println(mark);
```

✗

```
Create an account for Mark with balance 100:
Mark's current balance is: 100
Withdraw -1000000 from Mark's account:
Mark's current balance is: 1000100
```

Bank Accounts in Java: Version I Critique (3)



```java
public class BankAppV1 {
  public static void main(String[] args) {
    System.out.println("Create an account for Tom with balance 100:");
    AccountV1 tom = new AccountV1("Tom", 100);
    System.out.println(tom);
    System.out.println("Withdraw 150 from Tom's account:");
    tom.withdraw(150);
    System.out.println(tom);
```

```
Create an account for Tom with balance 100:
Tom's current balance is: 100
Withdraw 150 from Tom's account:
Tom's current balance is: -50
```