Lecture 6 - Sep. 26

Review on OOP, Exceptions

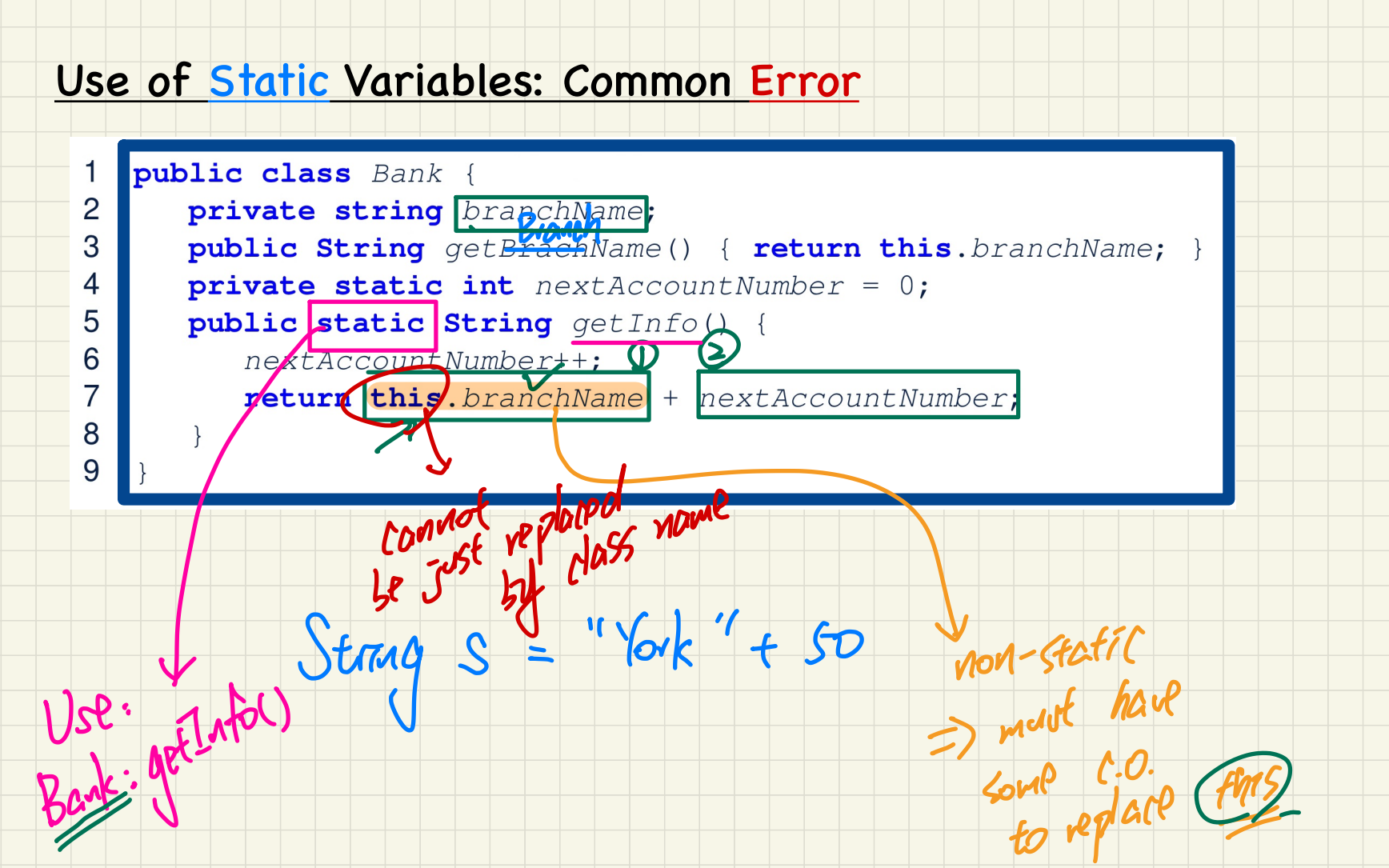
Static Variables: Common Error
Caller vs. Callee
Error Handling using Console Messages

Announcements

- Lab1 due at 2pm this Wednesday
- WrittenTest1
  - make sure you try logging into eClass in WSC
  - A guide and some practice questions released
- Programming Test 1 (60 to 65 min)
  - Identical format as Lab1
  - Number of starter tests will be smaller
  - Guide, Practice Test, Mockup Test to be announced

Use of Static Variables: Common Error

```
1  public class Bank {
2      private string branchName;
3      public String getBranchName() { return this.branchName; }
4      private static int nextAccountNumber = 0;
5      public static String getInfo() {
6          nextAccountNumber++;
7          return this.branchName + nextAccountNumber;
8      }
9  }
```



(solution 1)

static

```
1  public class Bank {
2      private string branchName;
3      public String getBrachName() { return this.branchName; }
4      private static int nextAccountNumber = 0;
5      public static String getInfo() {
6          nextAccountNumber++;
7          return this.branchName + nextAccountNumber;
8      }
9  }
```

non-static branch name ⇒ local. each Bank obj has its own branch

static branch name ⇒ global. all Bank objects share the same branch

↳ doesn't make real sense

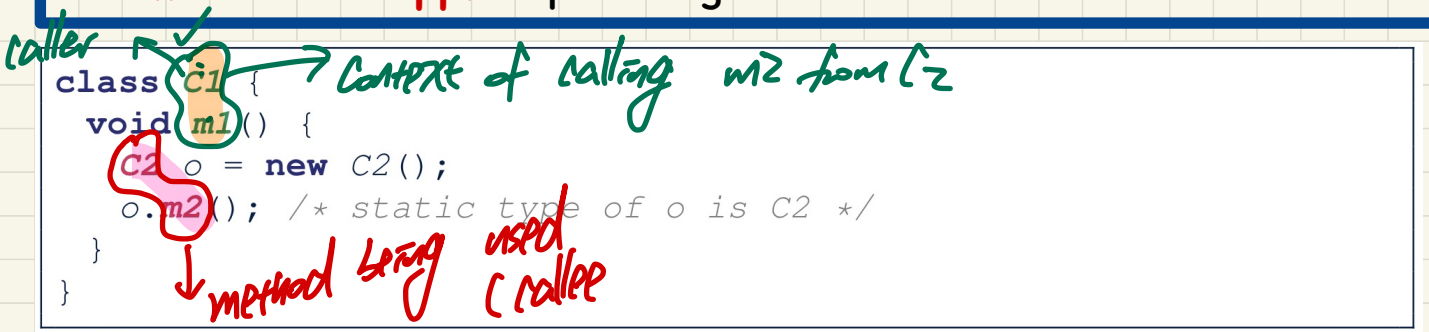
(solution 2).

```
1  public class Bank {
2      private string branchName;
3      public String getBrachName() { return this.branchName; }
4      private static int nextAccountNumber = 0;
5      public static String getInfo() {
6          nextAccountNumber++;
7          return this.branchName + nextAccountNumber;
8      }
9  }
```

Exercise -

Caller vs. Callee

- caller is the client using the service provided by another method.
- callee is the supplier providing the service to another method.

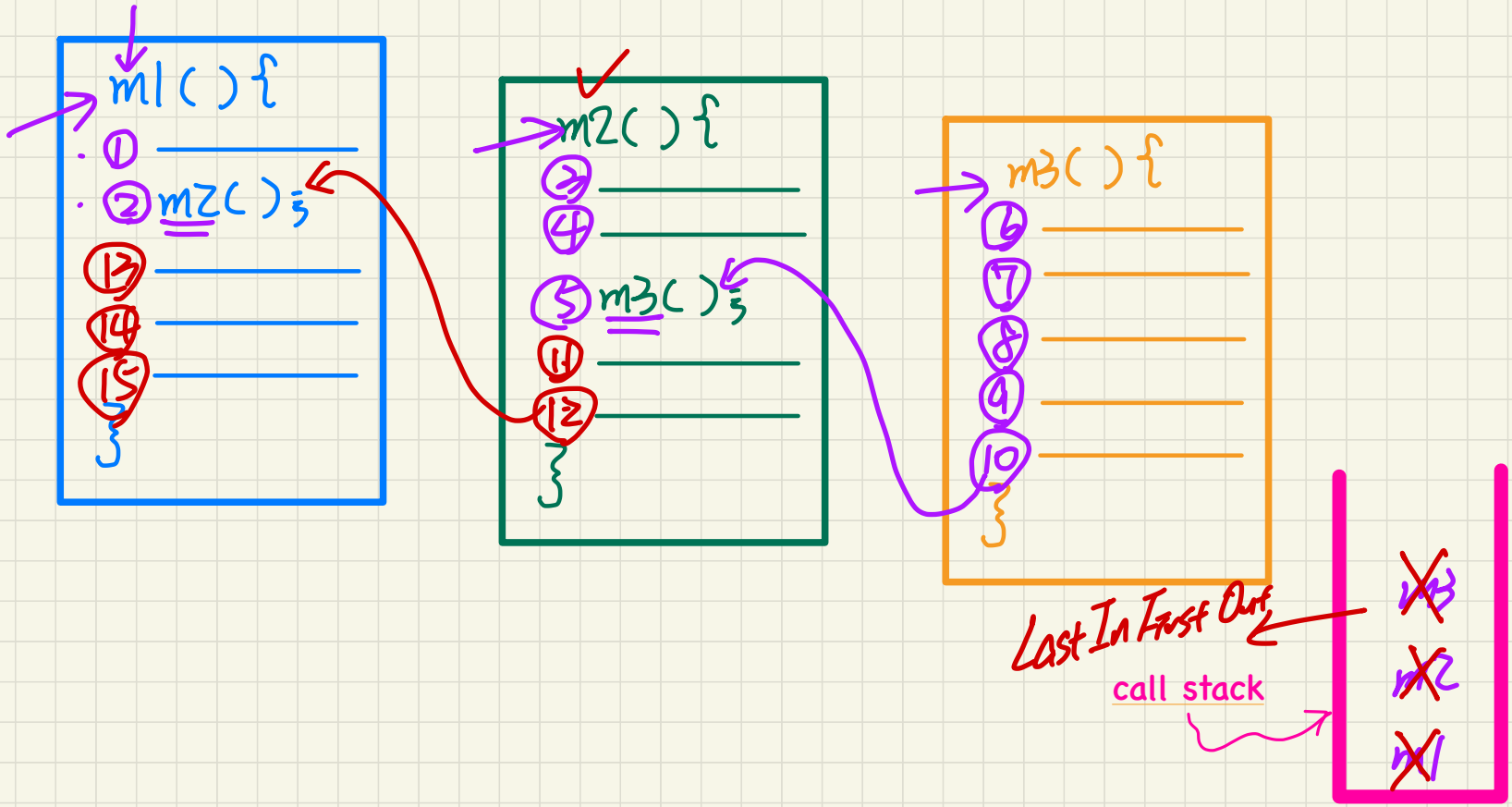


Q: Can a method be a caller and a callee simultaneously?

① class C3 {
void m3() {
C1 o = new C1(); o.m1();
} }

② class C2 {
void m2() {
C3 o = new C3();
o.m3();
} }

Visualizing a Call Chain using a Stack



Error Handling via Console Messages: Circles

```
1  class Circle {
2    double radius;
3    Circle() { /* radius defaults to 0 */ }
4    void setRadius(double r) {
5      if (r < 0) { System.out.println("Invalid radius."); }
6      else { radius = r; }
7    }
8    double getArea() { return radius * radius * 3.14; }
9  }
```

-10 -10

exits; can disrupt L5 & C6

Caller?

Callee?

call stack

```
1  class CircleCalculator {
2    public static void main(String[] args) {
3      Circle c = new Circle();
4      c.setRadius(-10);
5      double area = c.getArea();
6      System.out.println("Area: " + area);
7    }
8  }
```

to continue but it would not allow callee to handle the error (e.g. enter another #).

print error but would not stop the execution of L5 & C6

CircleCalculator.setRadius

CC.main

for error handling to be acceptable, these lines should not be allowed to continue.

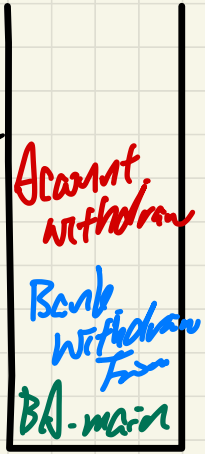
Error Handling via Console Messages: Banks

```
class Account {
  int id; double balance;
  Account(int id) { this.id = id; /* balance defaults to 0 */ }
  void deposit(double a) {
    if ( a < 0 ) { System.out.println( "Invalid deposit." ); }
    else { balance += a; }
  }
  void withdraw(double a) {
    if ( a < 0 || balance - a < 0 ) {
      System.out.println( "Invalid withdraw." ); }
    else { balance -= a; }
  }
}
```

Caller?

Callee?

call stack




```
class Bank {
  Account[] accounts; int numberOfAccounts;
  Bank(int id) { ... }
  void withdrawFrom(int id, double a) {
    for(int i = 0; i < numberOfAccounts; i ++) {
      if(accounts[i].id == id) {
        accounts[i].withdraw( a );
      }
    }
  }
}
```

```
class BankApplication {
  pubic static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    Bank b = new Bank(); Account acc1 = new Account(23);
    b.addAccount(acc1);
    double a = input.nextDouble();
    b.withdrawFrom(23, a );
    System.out.println("Transaction Completed.");
  }
```

context	caller	callee
BA	main	Bank.withdrawFrom
Bank	withdrawFrom	Account.withdraw
Account	withdraw	N.A.

Practice Written Test 1

Assume that a Person class is already defined, and it has an attribute name, a constructor that initializes the person's name from the input string, and an accessor `getName` returning the person's name. Consider the following fragment of Java code (inside some main method):

```
Person p0 = new Person("Suyeon");
Person p1 = new Person("Yuna");
Person p2 = new Person("Sunhye");
Person p3 = new Person("Jihye");
p0 = p2;
p1 = p3;
Person[] persons1 = {p0, p1, p2, p3};
Person[] persons2 = new Person[persons1.length];
for(int i = 0; i < persons2.length; i ++) {
    persons2[i] = persons1[persons2.length - i - 1];
}
```

Executing the above fragment of code, after exiting from the loop, indicate the value of each of the following expressions.

persons2[0].getName()	Choose...
persons2[1].getName()	Choose...
persons2[2].getName()	Choose...
persons2[3].getName()	Choose...