

Computer Science 2

Lecture 1: Part 2

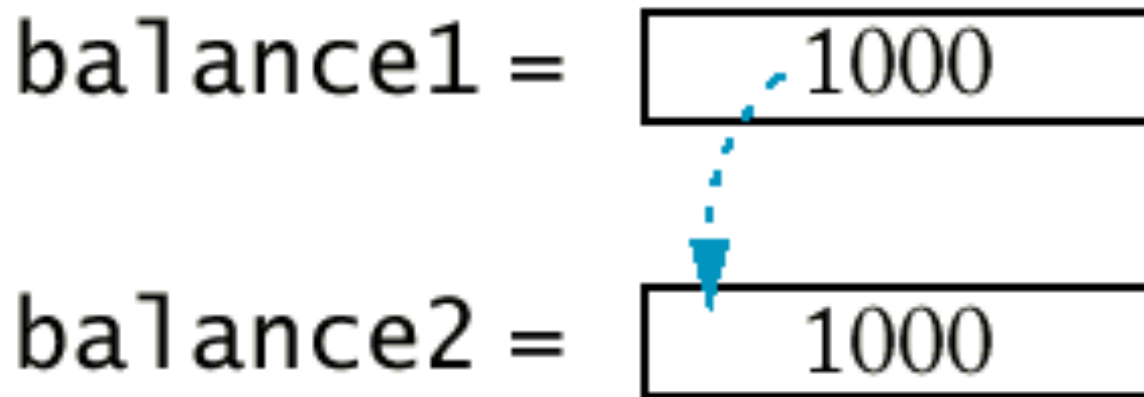
Non-Reference and Reference Variables, Static Fields, and Static Methods

Difference between Non-Reference Variables and Reference Variables

- Non-reference variables hold values of the same type;
- Reference variables hold references to objects, not the objects themselves;
- You can see the difference when you copy a variable.

Copying Numbers

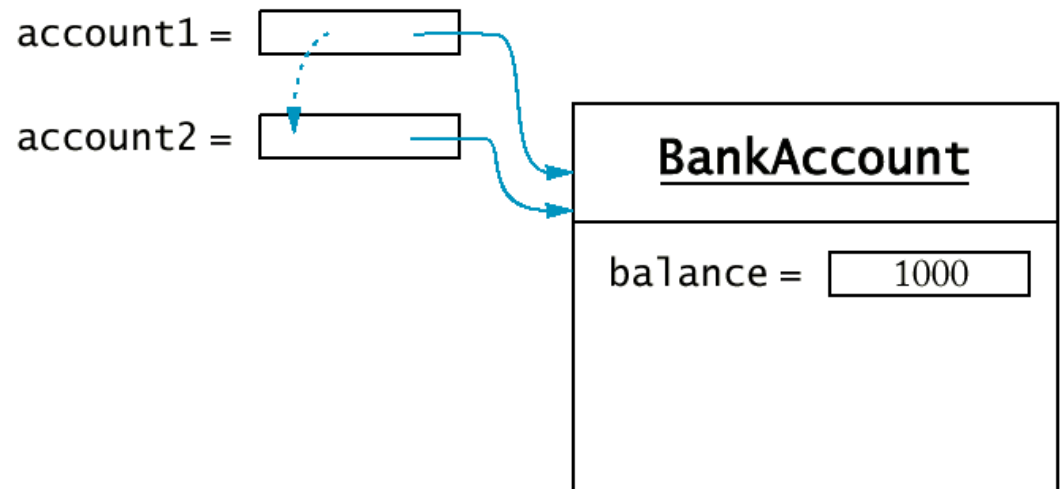
- `double balance1 = 1000;`
`double balance2 = balance1;`
`balance2 = balance2 + 500;`
- Change in `balance2` does not affect `balance1`



Copying Object References

```
BankAccount account1 = new BankAccount(1000);  
BankAccount account2 = account1;  
account2.deposit(500);
```

- Change through **account2** is also visible through **account1**



Non-Reference Variables in Method Calls

- The way how the factual values are passed to methods in Java is called *call-by-value*: for each parameter there is an expression, the expressions are evaluated, and the values of the expressions are passed as actual values.
- Thus, if a variable is passed to a method, the stored value of that variable in the calling method won't be changed.

```
int n = 5;  
do(n) ;  
System.out.println(n) ;
```

```
int n = 5;  
do(n + 10) ;  
System.out.println(n) ;
```

```
static void do(int k)  
{ k = k + 10 ; }
```

Reference Variables in Method Calls

- If reference variables are passed to a method, the stored values of the variables (references to objects) won't be changed. But, we can use the references inside the method to change the instance fields of the object.

```
BankAccount myAccount = new BankAccount ();  
System.out.print(myAccount.getBalance()); // prints 0.0  
changeObject(myAccount);  
System.out.print(myAccount.getBalance()); // prints 0.05
```

```
static void changeObject(BankAccount b)  
{ p.deposit(0.05); }
```

Static Fields

- **Static (Class) fields** belongs to the class, not to any object of the class;
- Static fields can be considered as fields that belong to all the objects of the class (black board)!

```
public class BankAccount
{   public BankAccount()
    {   balance = 0;
        lastAssignedNumber++;
        accountNumber = lastAssignedNumber;
    }

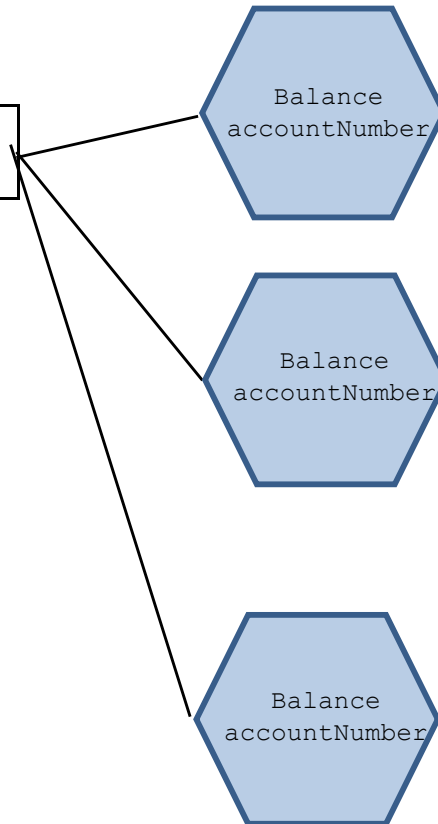
    private static int lastAssignedNumber;
    private int accountNumber;
    private double balance;
}
```

lastAssignedNumber

Balance
accountNumber


Balance
accountNumber

Balance
accountNumber



Static Fields

- There are three ways to initialize a static field:
 - Do nothing (the default value is assigned);
 - Use an explicit initializer;
 - Use a static initialization block.



```
public class BankAccount
{
    .....
    static
    {
        lastAssignedNumber = 0;
    }
    private static int lastAssignedNumber;
    private int accountNumber;
    private double balance;
}
```

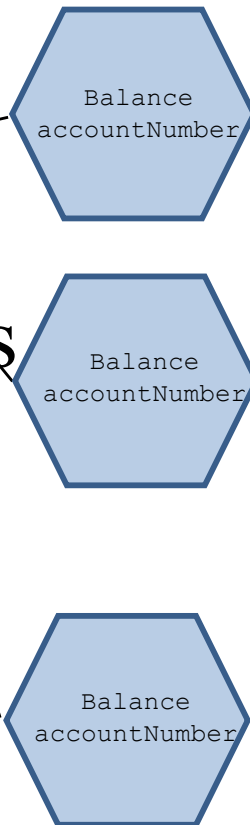
Static and Instance Fields

Static (Class) fields belongs to the class, not to any object of the class;

Static fields can be considered as fields that belong to all the objects of the class (black board)!

`lastAssignedNumber`

Instance fields belongs to the object of the class;
Instance fields are **not** shared between the objects of the class!



Static Methods

Static (Class) methods are those methods that do not have an implicit parameter; i.e., they do not operate on objects;

To define a static method, use the keyword **static**;

To call a static method, use the name of the class and the name of the method:

<Class Name>.<Method Name>(<parameter List>)

Example: **Math.sqrt(4.5);**

Example

```
public class Numeric
{   public static boolean equal(double x, double y)
    {
        final double EPSILON = 1E-14;
        if (x == 0) return Math.abs(y) <= EPSILON;
        if (y == 0) return Math.abs(x) <= EPSILON;
        return Math.abs(x - y) /
                Math.max(Math.abs(x), Math.abs(y))
                <= EPSILON;
    }
}
```

Too many static methods are a sign of too little OO!

Static and Instance Fields and Methods

Rules:

- Instance and static fields can be used from non-static methods;
- Static fields can be used from static methods;
- **Instance fields cannot be used from static methods.**

Static and Instance Fields, and Static and Instance Methods

```
public class BankAccounts
{
    public BankAccounts()
    {
        lastAssignedNumber++;
        accountNumber = lastAssignedNumber;
    }
    public void withdraw(double amount)
    {
        balance = balance - amount;
    }
    public static int getNumberAccounts()
    {
        return lastAssignedNumber;
    }

    .....
    private static int lastAssignedNumber = 0;
    private int accountNumber;
    private double balance = 0;
}
```

Concepts Covered in the Lecture

- Comparison of Non-Reference and Reference Variables.
- Static Fields
- Static Methods

