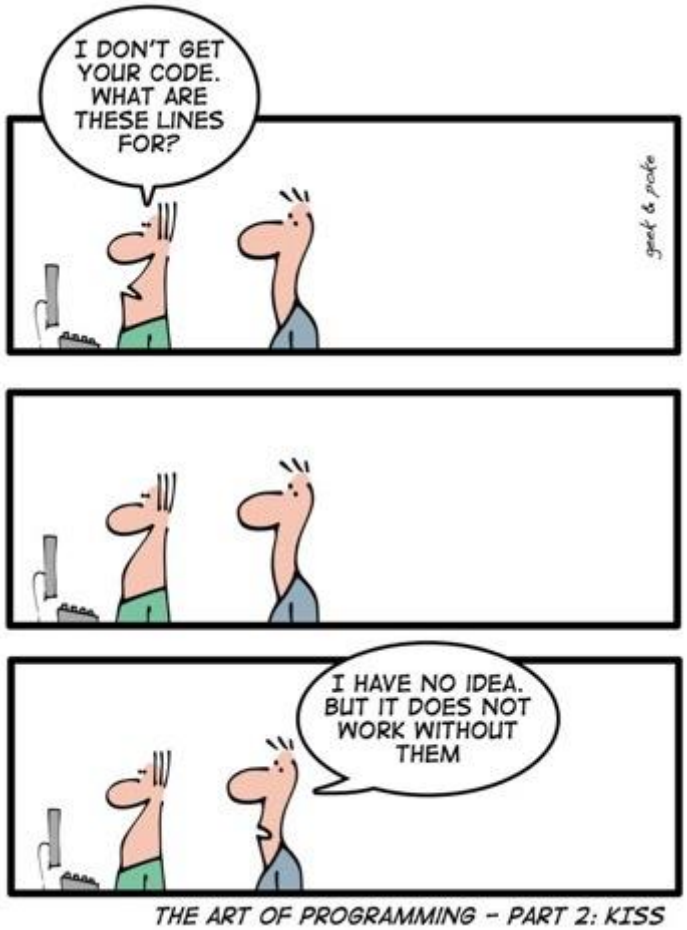
Java Class 11



How To Define A Method

A method declaration (definition) begins with a ***method header***

```
char calc (int num1, int num2, String message)
```

return type

method name

parameter list

The parameter list specifies the type and name of each parameter

The name of a parameter in the method declaration is called a *formal parameter*

Method Body

The method header is followed by the ***method body***

```
char calc (int num1, int num2, String message)
{
    int sum = num1 + num2;
    char result = message.charAt (sum);

    return result;
}
```

The return expression must be consistent with the return type

`sum` **and** `result` **are local data**

They are created each time the method is called, and are destroyed when it finishes executing

The return Statement

The ***return type*** of a method indicates the type of value that the method sends back to the calling location

A method that does not return a value has a `void` return type

A ***return statement*** specifies the value that will be returned

```
return expression;
```

Its expression must conform to the return type

Parameters

When a method is called, the ***actual parameters*** in the invocation are copied into the ***formal parameters*** in the method header

```
int count = 2;
ch = obj.calc (5, count, "Hello World");
```

```
char calc (int num1, int num2, String message)
{
    int sum = num1 + num2;
    char result = message.charAt (sum);
    return result;
}
```

Scope and Local Data

As we've seen, local variables can be declared inside a method

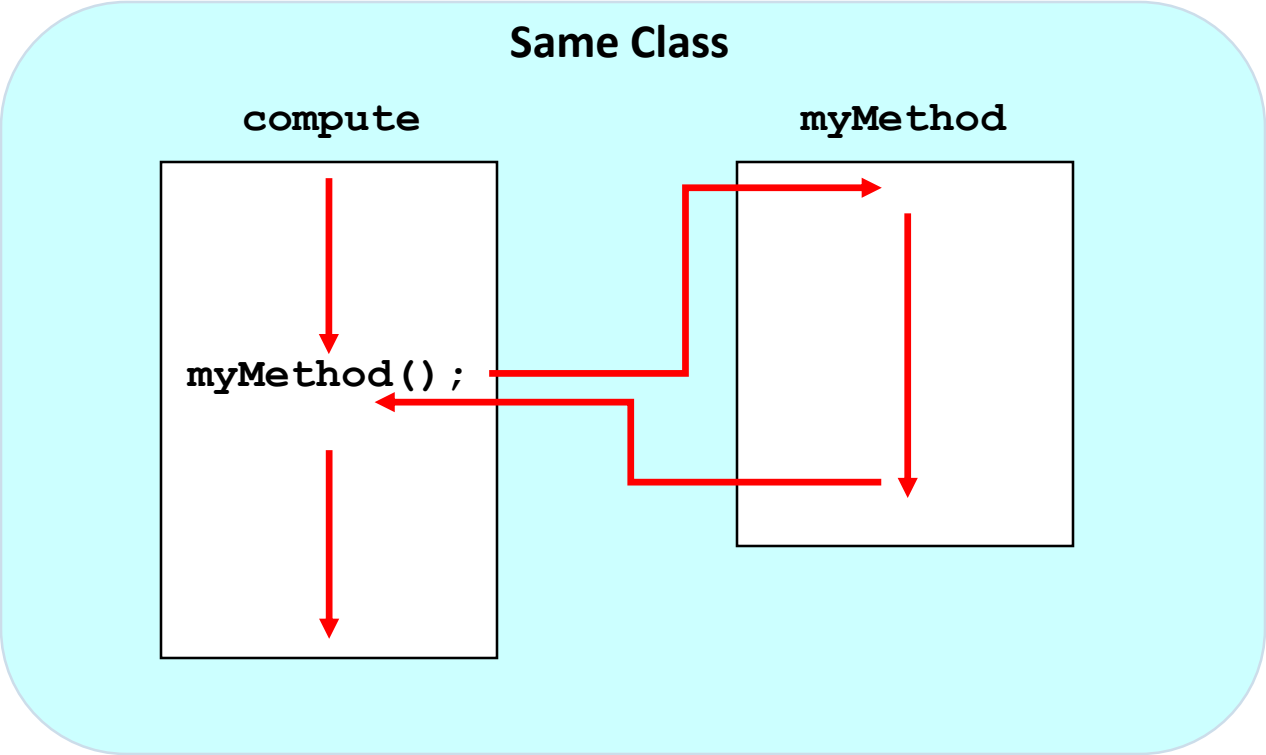
Keep in mind that instance variables, declared at the class level, exist as long as the object exists

The formal parameters of a method create *automatic local variables* when the method is invoked

When the method finishes, all local variables are destroyed (including the formal parameters)

Method Control Flow

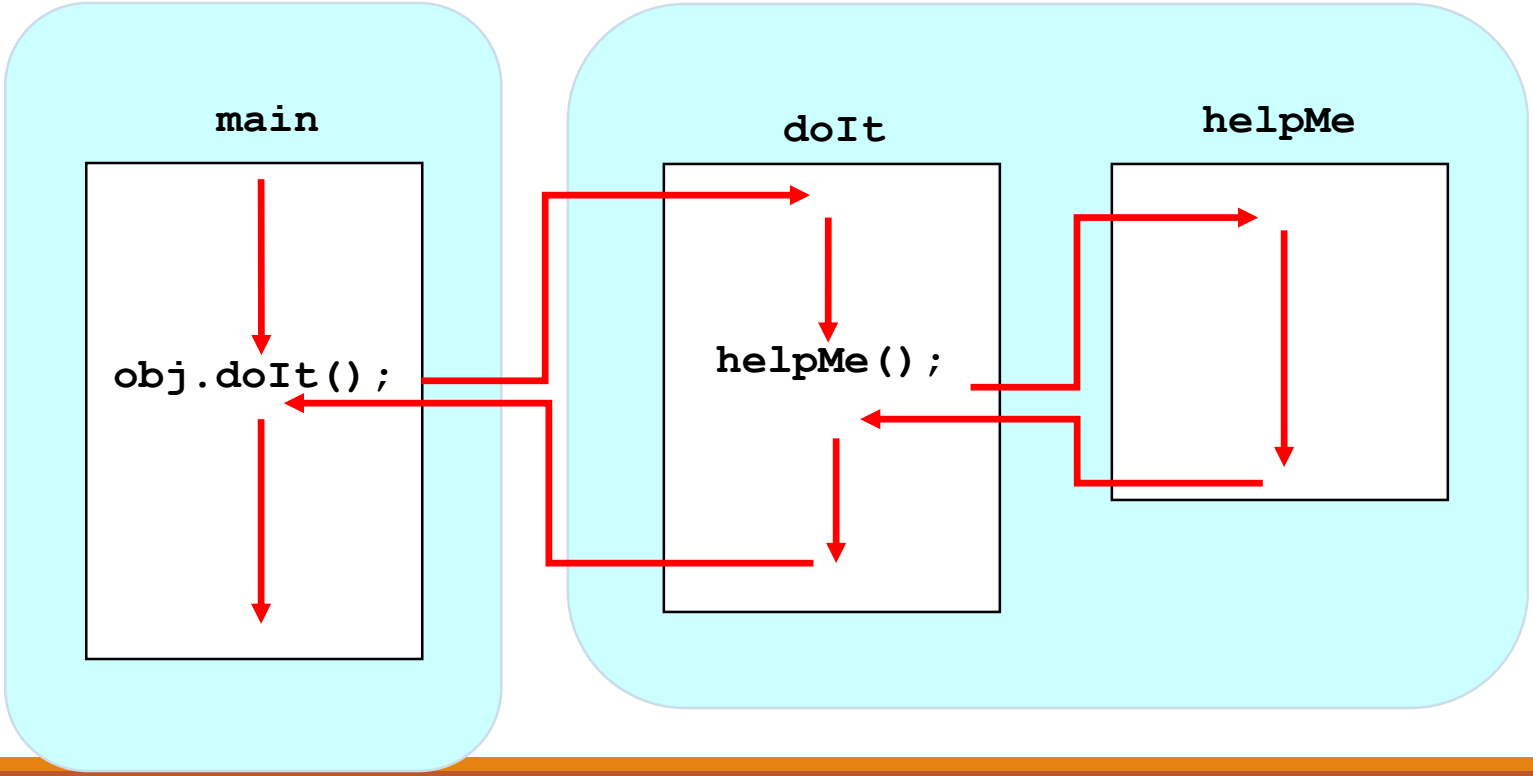
If the called method is in the same class, only the method name is needed



Method Control Flow

The called method is often part of another class or object. When that happens, the method is called using the class or object name, dot operator, and the method name

Two Separate Classes



Constructors Revisited

```
Duck myDuck = new Duck();
```

It looks like we're calling a method named Duck(), because of the parentheses.

Are we calling a method named `Duck`?

Not really, we are calling the Duck constructor

The constructor runs when you instantiate an object when using the keyword `new` followed by the class name

If a class has no constructor (such as the `Dog` class), the compiler automatically creates a default constructor

```
public Dog() {

}
```

← default constructor

Constructors

How is a constructor different from a regular method?

- The name of a constructor is the same as the class name
- A constructor cannot return a value and does not have a return type (it is even not a `void` return type)

A common usage of a constructor: to initialize the state (instance variables) of an object with parameters (arguments)

```
public class Die {
   private int faceValue;

   // constructor with arguments
   public Die(int value) {
     faceValue = value;
   }
}
```

To make a Die with faceValue 3

```
Die d = new Die(3);
```

Constructors

```
public class Account {

   private long acctNumber;
   private double balance;
   private String name;

   // constructor with arguments
   public Account (String owner, long account, double initial) {
      name = owner;
      acctNumber = account;
      balance = initial;
   }
}
```

To create Account objects with initial values

```
Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
Account acct2 = new Account ("Jane Smith", 69713, 40.00);
Account acct3 = new Account ("Edward Demsey", 93757, 759.32);
```

Constructors

1. A constructor is the code that runs when somebody says **new** on a class type

```
Duck d = new Duck();
```

2. A constructor must have the same name as the class, and ***no*** return type

```
public Duck(int size) { }
```

3. If you don't put a constructor in your class, the compiler puts in a default constructor. The default constructor is always a no-arg constructor.

```
public Duck() { }
```

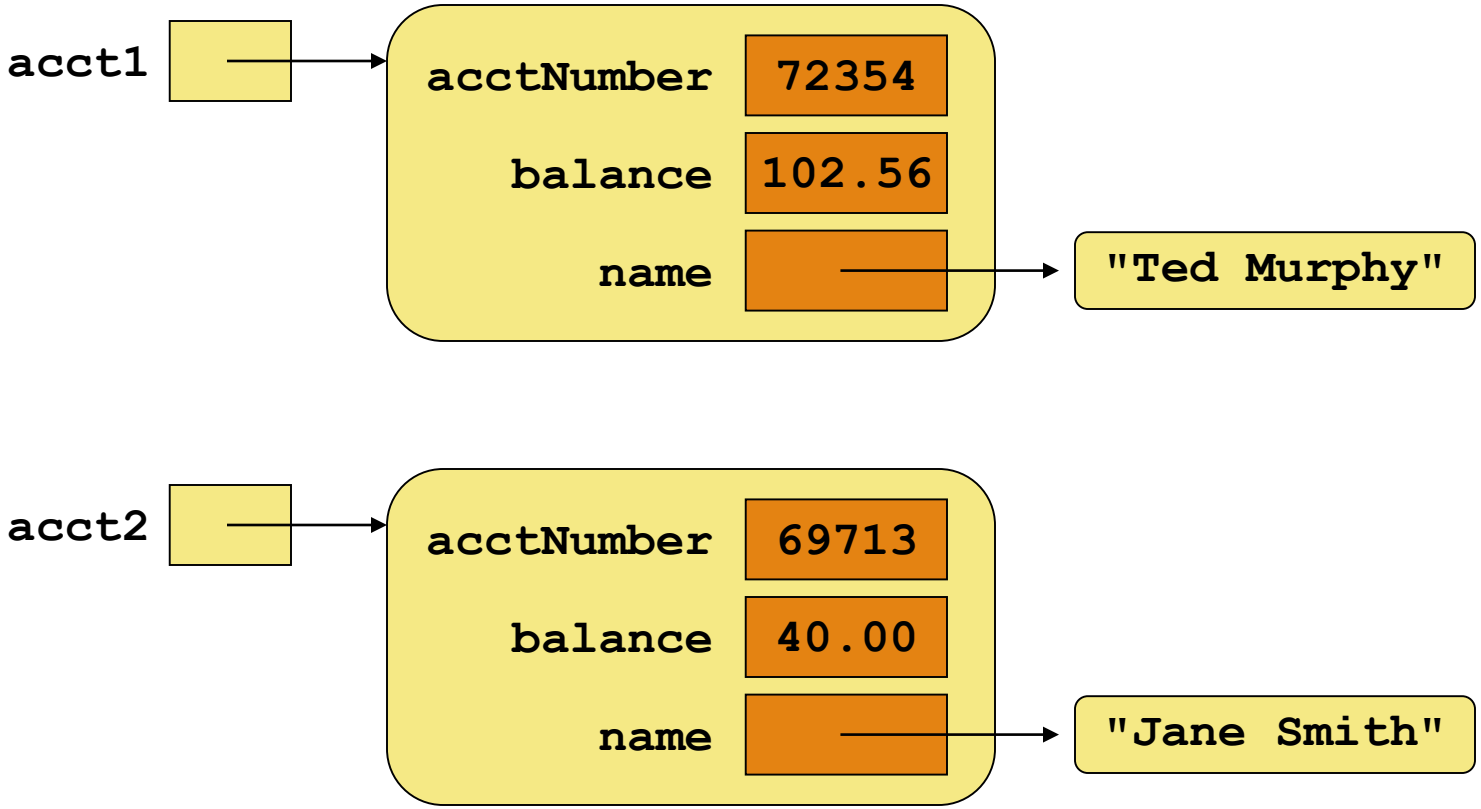
Driver Programs

A ***driver program*** drives the use of other, more interesting parts of a program

Driver programs are often used to test other parts of the software

The `Transactions` class contains a `main` method that drives the use of the `Account` class, exercising its services

Bank Account Example



```java
//*******************************************************************
//  Transactions.java       Author: Lewis/Loftus
//
//  Demonstrates the creation and use of multiple Account objects.
//*******************************************************************

public class Transactions
{
   //-----------------------------------------------------------------
   //  Creates some bank accounts and requests various services.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
      Account acct2 = new Account ("Jane Smith", 69713, 40.00);
      Account acct3 = new Account ("Edward Demsey", 93757, 759.32);

      acct1.deposit (25.85);

      double smithBalance = acct2.deposit (500.00);
      System.out.println ("Smith balance after deposit: " +
                          smithBalance);
```

continue

continue

```
      System.out.println ("Smith balance after withdrawal: " +
                          acct2.withdraw (430.75, 1.50));

      acct1.addInterest();
      acct2.addInterest();
      acct3.addInterest();

      System.out.println ();
      System.out.println (acct1);
      System.out.println (acct2);
      System.out.println (acct3);
   }
}
```

continue

```
        System.                                                    +

        acct1.a
        acct2.a
        acct3.a

        System.out.println ();
        System.out.println (acct1);
        System.out.println (acct2);
        System.out.println (acct3);
    }
}
```

Output

```
Smith balance after deposit: 540.0
Smith balance after withdrawal: 107.55

72354   Ted Murphy      $132.90
69713   Jane Smith      $111.52
93757   Edward Demsey   $785.90
```

```java
//*******************************************************************
//  Account.java       Author: Lewis/Loftus
//
//  Represents a bank account with basic services such as deposit
//  and withdraw.
//*******************************************************************

import java.text.NumberFormat;

public class Account
{
   private final double RATE = 0.035;  // interest rate of 3.5%

   private long acctNumber;
   private double balance;
   private String name;

   //-----------------------------------------------------------------
   //  Sets up the account by defining its owner, account number,
   //  and initial balance.
   //-----------------------------------------------------------------
   public Account (String owner, long account, double initial)
   {
      name = owner;
      acctNumber = account;
      balance = initial;
   }
```

continue

continue

```
//-----------------------------------------------------------------
//  Deposits the specified amount into the account. Returns the
//  new balance.
//-----------------------------------------------------------------
public double deposit (double amount)
{
   balance = balance + amount;
   return balance;
}

//-----------------------------------------------------------------
//  Withdraws the specified amount from the account and applies
//  the fee. Returns the new balance.
//-----------------------------------------------------------------
public double withdraw (double amount, double fee)
{
   balance = balance - amount - fee;
   return balance;
}
```

continue

continue

```
   //-----------------------------------------------------------------
   //  Adds interest to the account and returns the new balance.
   //-----------------------------------------------------------------
   public double addInterest ()
   {
      balance += (balance * RATE);
      return balance;
   }

   //-----------------------------------------------------------------
   //  Returns the current balance of the account.
   //-----------------------------------------------------------------
   public double getBalance ()
   {
      return balance;
   }

   //-----------------------------------------------------------------
   //  Returns a one-line description of the account as a string.
   //-----------------------------------------------------------------
   public String toString ()
   {
      NumberFormat fmt = NumberFormat.getCurrencyInstance();
      return (acctNumber + "\t" + name + "\t" + fmt.format(balance));
   }
}
```

Quick Check

How do we express which `Account` object's balance is updated when a deposit is made?

Each account is referenced by an object reference variable:

```
Account myAcct = new Account(…);
```

When a method is called, you call it through a particular object:

```
myAcct.deposit(50);
```

Group Exercises

Ex: 4.6

Ex: 4.7

Assignment for Class 12

Review Transactions, Account

Read Chapter 4.6, 4.7, 4.8, 5.7