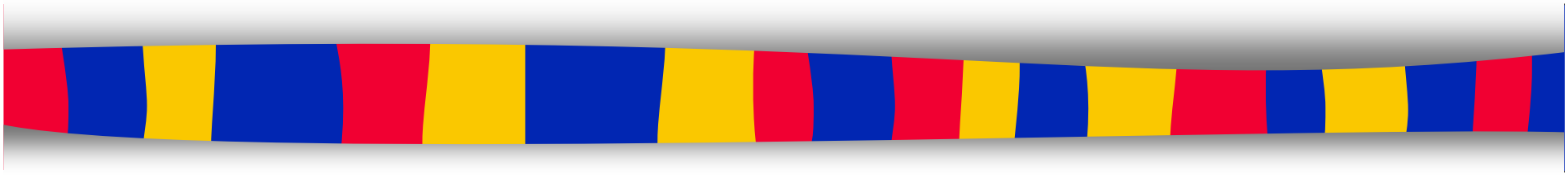
COMP-248
Object Oriented Programming I



Classes and Objects

By Emad Shihab, PhD, Fall 2015,
Parts of the slides are taken from Prof. L. Kosseim
Adapted for Section EE by S. Ghaderpanah, Fall 2015

Classes and Objects

- A class is a type and you can declare variables of a class type.

- A value/instance of a class is called an objects. An object has 2 components:

    - *Data (instance variables)* - descriptive characteristics

    - *Actions (methods)* - what it can do (or what can be done to it)

Example: A class definition

```
public class Date
{
   public String month;
   public int day;
   public int year;

   public boolean isWeekEnd ()
   {
      …
   }

   public void printDate()
   {
      ...
   }
}
```

data declarations (instance variables)

action declarations (methods)

Declaring Objects

- The new operator is used to create an object of a class and associate it with a variable

- Example:
  - `nameOfClass nameOfObject = new nameOfClass();`
  - `nameOfClass nameOfObject ;`
    `nameOfObject = new nameOfClass();`

Example: A driver file

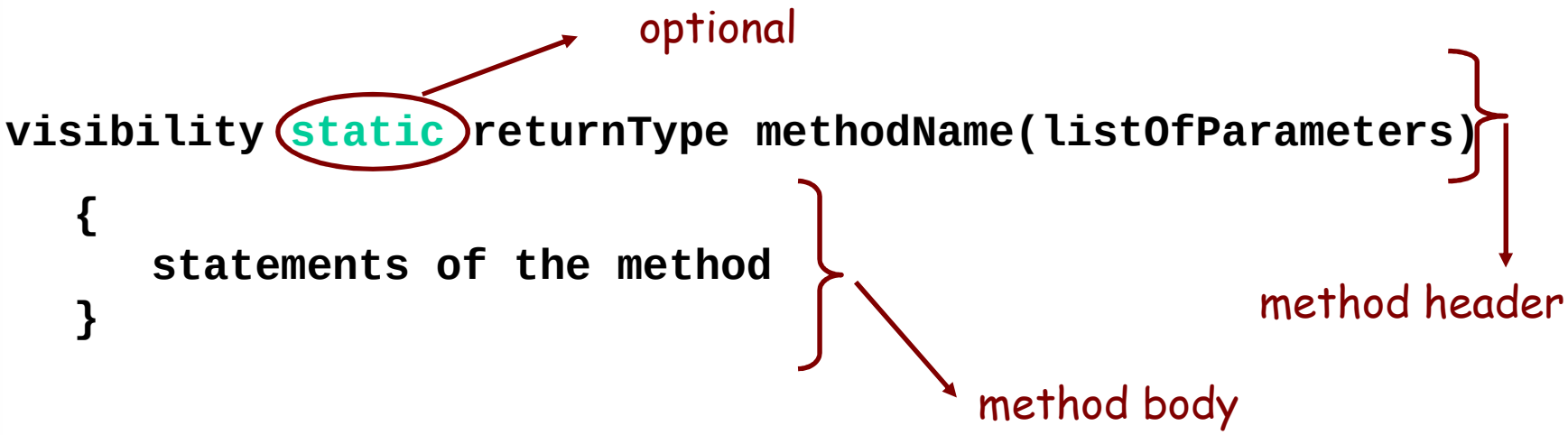
```
public class DateFirstTryDemo
{
   public static void main(String[] args)
   {
      Date date1;
      date1 = new Date();
      Date date2 = new Date();

      date1.month = “December”;
      date1.day = 31;
      date1.year = 2012;
      date1.printDate( );

   }
}
```

declaration of 2 objects

usage of the object

Definition of a method

- Method header and method body




```
public void sayHi()
{
  System.out.print("Hi");
}
```

```
public static void main(String[] args)
{
   …
}
```

Methods cont'd…

There are two kinds of methods:

- Methods that compute and return a value
- Methods that perform an action
  does not return a value
  is called a **`void`** method

Today we will see:

1. Writing our own classes
   1.1 Classes and Objects
   1.2 Instance Variables
   1.3 **Methods (More)**
2. Some notions of OOP
3. Passing and returning objects
4. Recap

Accessing Class Members

To access any members (data or method)

- within the class

```
nameOfData
nameOfMethod(actualParameters)
```

- from outside the class
  - non-static:

```
nameOfObject.nameOfData
nameOfObject.nameOfMethod(actualParameters)
```

  - Static:

```
We will cover this late…
```

Example

```
public class Coin
{
   public final int HEADS = 0;
   public final int TAILS = 1;
   public int face;

   public void flip()  {
      face = (int)(Math.random()*2);
   }

   public boolean isHeads() {
      return (face == HEADS);
   }

  // flips the coin 5 times
   public void flip5() {
      for (int i=1; i<=5; i++)
         flip();
   }
…
}
```

class def.

Example

```
public class Coin
{
   public final int HEADS = 0;
   public final int TAILS = 1;
   public int face;


   public void flip()  {

      face = (int)(Math.random()*2);

   }



   public boolean isHeads() {

      return (   face    == HEADS);

   }


  // flips the coin 5 times
   public void flip5() {

      for (int i=1; i<=5; i++)
         ____________flip();____________;

   }
}
```

class def.

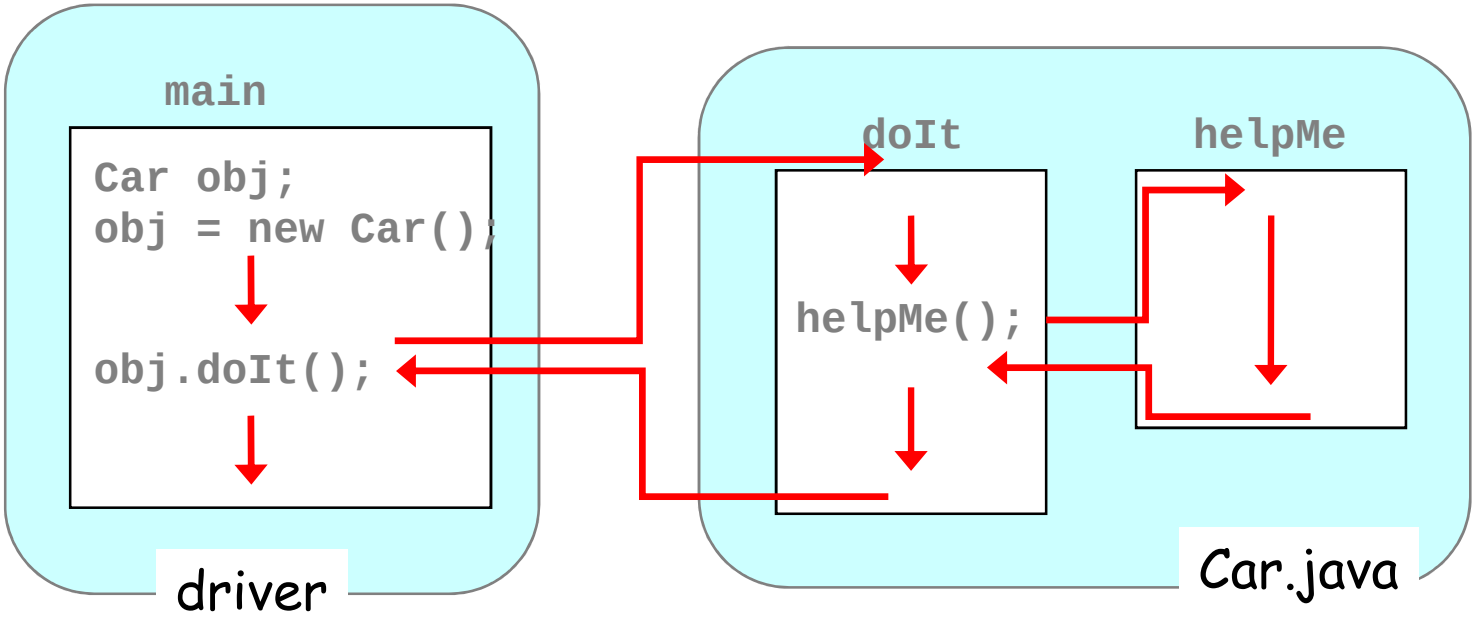
```
public class FlipRace
{
   public static void main (String[] args)
   {
      final int GOAL = 3;
      int count1 = 0, count2 = 0;

      // Create two separate coin objects
      Coin coin1 = new Coin();
      Coin coin2 = new Coin();

      // Flip the coins and count heads
      while (count1<GOAL && count2<GOAL) {
         ____________coin1.flip()____________;

         ____________coin2.flip()____________;

         if (coin1.isHeads()) count1++;

         if (coin2.isHeads()) count2++;

      }

      flip();  // 1. OK?

      coin1.flip;  // 2. OK?
   }
}
```

driver

Method Control Flow

When a method is invoked:

1. the current method is suspended
2. the called method is executed
3. when completed, the flow returns to the place where the method was called and resumes its execution



Calling Methods

Methods that return a result are expressions that have:

a type and a value

Methods that do not return a result are **statements**

you call them with the statement: `objectName.methodName();`

```
public class Driver {
   public static void main (String[] args)   {
      Coin coin1 = new Coin();
      boolean result;

      coin1.flip();           // 1. OK?   Yes
      result = coin1.flip(); // 2. OK?   No
      if (coin1.flip())      // 3. OK?   No
         System.out.print("whatever");
      System.out.print(coin1.flip()); //4. OK?  No

   }
}
```

Just checking ....

Which one of these statements is syntactically correct?

```
        Coin coin1 = new Coin();
        boolean result;
/* 1 */ coin1.isHeads();
/* 2 */ result = coin1.isHeads();
/* 3 */ if (coin1.isHeads())
         System.out.print("whatever");
/* 4 */ System.out.print(coin1.isHeads());
```

A. All are syntactically correct
B. 2 and 3 only are syntactically correct
C. 2, 3 and 4 only are syntactically correct
D. 1 is the only one which is syntactically correct
E. 3 is the only one which is syntactically correct

Example:

DateSecondTry.java

DemoOfDateSecondTry.java

Add a method to the DateSecondTry class called getNextDay that returns an int corresponding to the day+1

How would you modify DemoOfDateScond to print out the year

Just checking …

The body of a method that returns a value must contain at least one _____ statement.

A. void
B. invocation statement
C. declaration
D. return

Local / Instance/ Global Variables

Local variables:

- Declared inside a method
  - variables that are necessary for that method only
  - method parameters are considered local variables
- If 2 methods have a local variable of the same name, they are 2 entirely different variables

Instance variables:

- Confined to an object of the class

Global variables:

Java does **not** have global variables

Parameters

- Some methods need to receive additional data in order to perform their work

- Allows the function to be more generic

  ex. **`sqrt`** method works for any double

```
Math.sqrt(9), Math.sqrt(15.5),
  Math.sqrt(75),...
```

Definitions:

Formal parameter:

parameter specified in the method header

Argument:

The value that is plugged in for the parameter

Example 1

```
public class Account
{
   private double balance;

   public void deposit (double theAmount)
   {
      if (theAmount < 0)  // deposit value is negative
         System.out.println ("Error: Deposit amount is invalid.");
      else
         balance = balance + theAmount;
}
```

class def.

```
public class Banking
{
   public static void main (String[] args)
   {
      Account acct1 = new Account();
      acct1.deposit(25.85);
      acct1.deposit(10);
   }
}
```

driver

Q: How would you modify the existing code so that the balance can be printed?

Example 2

```
public class Store
{
   public void printPrice(_____________________________ , _____________________________ )
   {
     System.out.println ("the " + ___________________ + " costs " + _________________);
   }
 …
```

class def.

```
public static void main (String[] args)
{
  Store myStore = new Store();
  myStore.printPrice("desk", 300.99);

  double special = 195.99;
  myStore.printPrice("chair", special);

  Store yourStore = new Store();
  double p = Keyboard.readDouble();
  String s = "desk";
  yourStore.printPrice(s, Math.min(100, p));
    …
}
```

driver

Exercise

in the class **Account,** write the header of the methods:

```
public class Account {
   private long acctNumber;
   private double balance;
   private String name;
   private double interestRate;
   …
     // withdraw
     // getInterestRate
     // changeName
   }
}
```

```
public void withdraw(int amount)
        {
                balance -= amount;
        }
```

```
public double getInterestRate()
        {
                return interestRate;
        }
```

```
public void changeName(String newName)
        {
                name = newName;
        }
```

Exercise cont'd

In a driver class… write instructions to create an account and call the previous methods

```
…

Account acct1 = new Account();
acct1.withdraw(50);
double x = acct1.getInterestRate();
acct1.changeName("Emad");
System.out.println(x);
```

Just checking …

A variable whose meaning is confined to a method definition is called an/a

A. instance variable
B. local variable
C. global variable
D. none of the above

Next:

1. Writing our own classes
   - 1.1 Classes and Objects
   - 1.2 Instance Variables
   - 1.3 **Methods (More)**
2. **Some notions of OOP**
3. Passing and returning objects
4. Recap