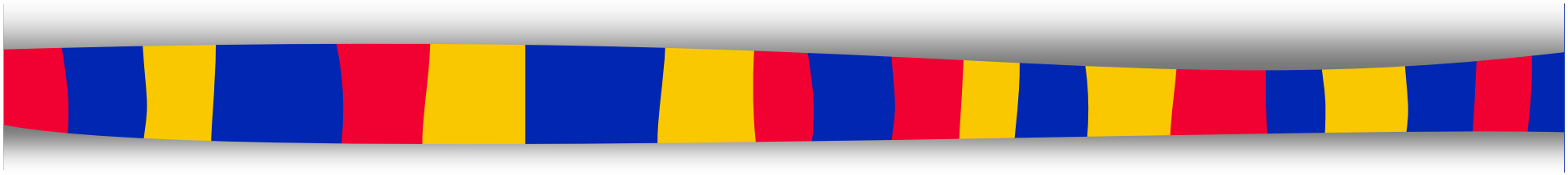
COMP-248
Object Oriented Programming I



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

In this chapter, we will see:

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

Predefined classes

- Programmers can:
  use pre-defined classes
  develop their own classes

- a *class library* is a collection of pre-defined classes

- the *Java standard class library* (or *Java API* - Application Programming Interface) defines many classes
  ex:
  the **`System`** class
  the **`String`** class

Writing our own classes…

So far, our program had only 1 class with 1 method (**main**)

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

For large problems…

- the program becomes large and difficult to understand
  e.g., repetition of instructions, variables are dispersed…

Solution:

- Decompose the problem into sub-problems.
- Use **methods** to implement the sub-problems
- Group related variables and methods into **classes**

Classes and Objects

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

- A value 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)

Objects

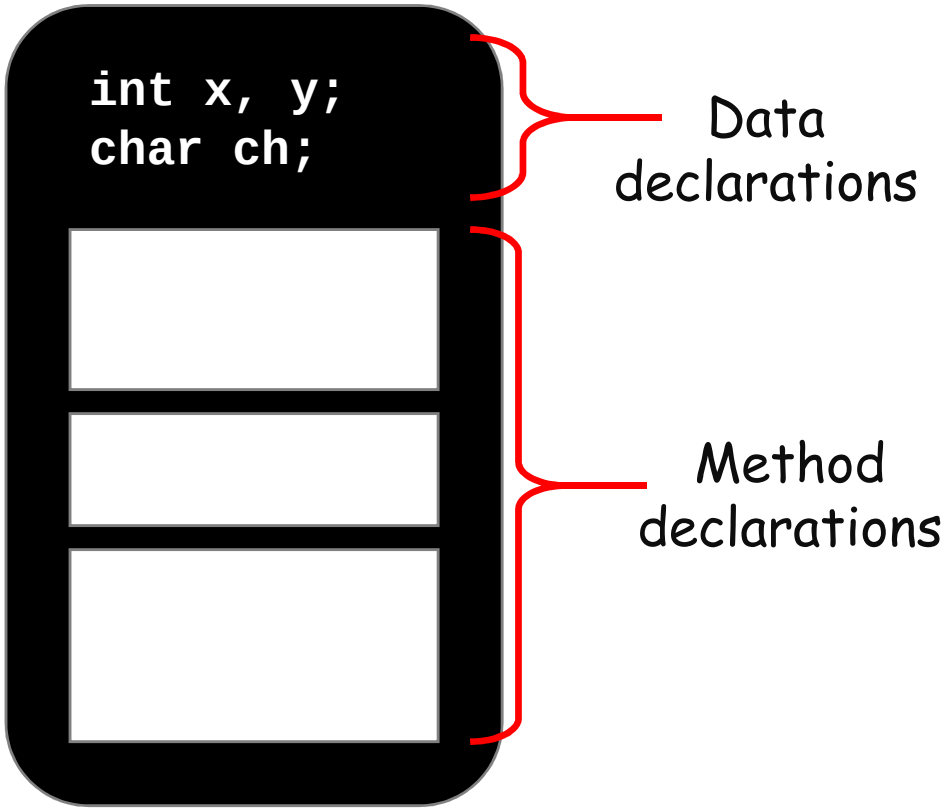
- An object is often referred to as an instance of the class

- Each object can have **different data**, but **all objects have the same actions**.

Declaration of Classes

A class contains:

1. data declarations (instance variables)
2. action declarations (methods)



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)

Exercise 1

- Write a class, Car, that can be used to create car objects.
- Each car has a make, model, color and year.
- Also create a printCar() that prints each of the car's instance variables.

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

Q: How many instance variable and methods does our object have?

Classes and files

- In general, we store each class in its own file
- Our example program has 2 files:

1. The **class definition file**:

   defines the data and methods of a class

2. The **driver file**:

   contains the **`main`** method

   declares and uses objects of the class
   controls the use of other parts of a program
   often used to test other parts of the program

Just checking …

The new operator:

A. allocates memory
B. is used to create an object of a class
C. associates an object with a variable that names it.
D. all of the above.

Exercise 1 cont'd …

- Write a driver class that declares 2 Car objects
- Set the instance variables for each car and use the printCar() method to print out the instance variables

In this chapter, we will see:

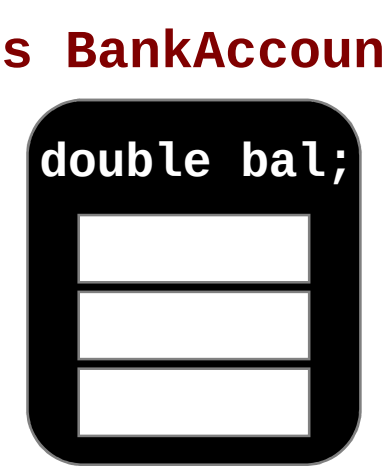
1. Writing our own classes
   - 1.1 Classes and Objects
   - 1.2 **Instance Variables**
   - 1.3 Methods
2. Some notions of OOP
3. Passing and returning objects
4. Recap

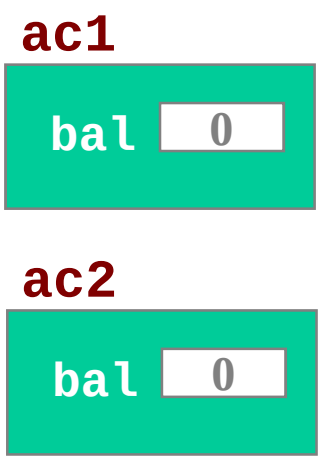
Instance Variables

- variables/constants declared inside the class **but not inside a specific method**
- also called attributes
- can be used by any method of the class
- initialized to 0 (false for booleans)
- each object (instance) of the class has its own instance data

Example:

```
BankAccount ac1=new BankAccount();
BankAccount ac2=new BankAccount();
```





In this chapter, we will see:

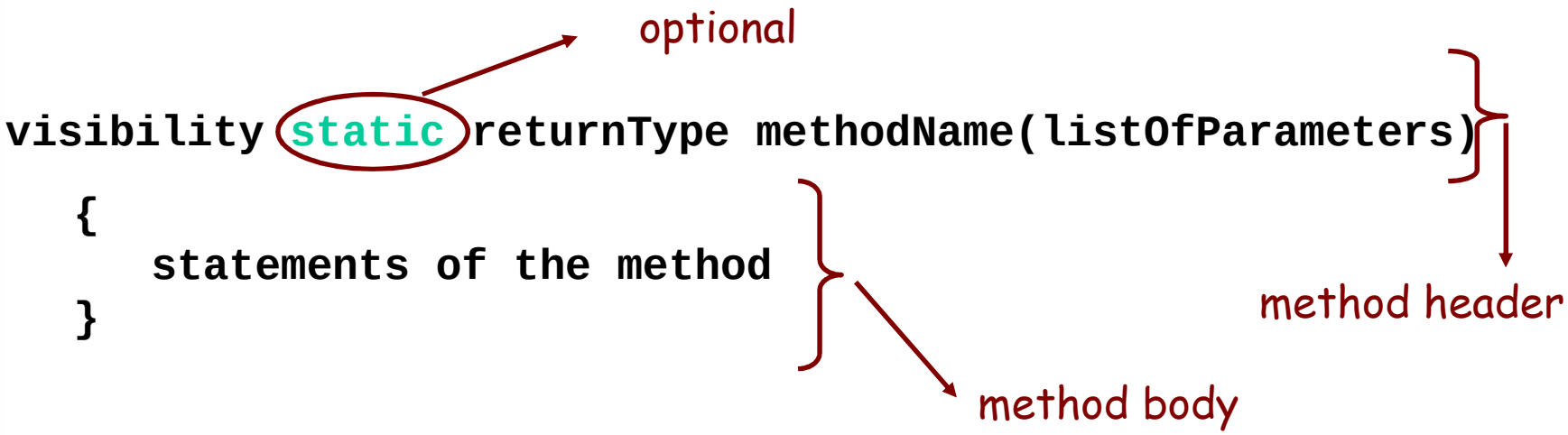
1. Writing our own classes
   - 1.1 Classes and Objects
   - 1.2 Instance Variables
   - 1.3 **Methods**
2. Some notions of OOP
3. Passing and returning objects
4. Recap

Methods

- Implement the behavior of all objects of a particular class
- All objects of a class share the method definitions
- Some methods are a bit special…
  (ex: constructor)
- Group of statements that are given a name
- A method is defined once, but can be used (called/invoked) several times

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

Methods that return a result

- Must specify the type of that value in its heading:
  `public typeReturned methodName(paramList)`

- Examples:

  description: determines if the coin is a tail (1==tail, 0==heads)
  name: **`isTail`**
  result: **`boolean`**

```
public boolean isTail()
{
  if (face == 1)
     return true;
  else
     return false;
}
```

  description: returns the value of the face
  name: **`getFace`**
  result: **`int`**

```
public int getFace()
{
   return (face);
}
```

The return statement

Allows the method to "return" a result

syntax: **`return expression;`**

1. the expression is evaluated
2. the value of the expression is returned as the result of the method
3. the method is exited

```
public boolean isTail()
{
  if (face == 1)
     return true;
  else
     return false;
}
```

```
public int getFace()
{
   return (face);
}
```

```
public boolean isTail()
{
  return (_____________________________);
}
```

Another way of writing the method **`isTail()`** is …. ?

Methods that return no result

they perform an action performed, but do not "evaluate" to a value

ex: they display something, change the value of an attribute, …

They officially return **void**

They use no **return expression;**

or: **return;**

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

```
public _____ printFace()
{
  System.out.print(face);
}
```

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

Exercise 1 cont'd …

- Modify your Car class to add a method called getYear() that returns the year of a given car object

Next class, 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