LESSON IV.
Encapsulation and Class Building

Vu Thi Huong Giang
SoICT (School of Information and Communication Technology)
HUST (Hanoi University of Science and Technology)

Objectives

- Understand the application of object oriented principles in Java
- Acquaint how to declare a class and its members

Content

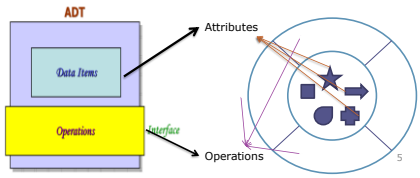
- Data abstraction
  - Overview
  - Class and instance
  - Message passing
- Encapsulation
  - Visibility
- Class building
  - Declaration
    - Class declaration
    - Class member declaration
  - Data hiding

Abstraction

- Abstraction hides the detailled information about object.
- Abstraction is a view or representation of an object that includes only the most significant attributes
  - These attributes make dicstinction this object with others.

Abstract data type

- Abstract data type = data type + operation performed on this type
  - Every operation related to a data structure is grouped together
  - Only possible operation are provided in the type's definition
- Java allows implementing ADT in the form of classes.
- A class comprises of attributes and operations.



2. Class

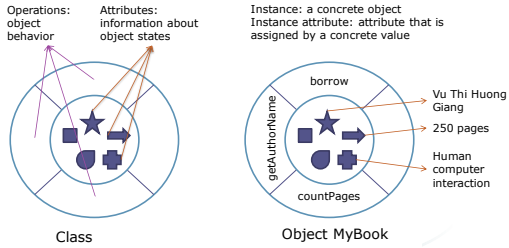
- A class is the blueprint from which individual objects are created.
- A class specifies the common attributes and operations of many individual objects all of the same kind.
- A class defines a new data type, whose values are objects:
  - A class is a template for objects; it abstracts a set of objects
  - An object is an instance of a class; it concretes a class.

Attributes and operations

- Attribute
  - An attribute of a class is an identified common state of all instances of this class (i.e. set of concrete objects).
  - An attribute specifies all possible values that can be assigned as concrete state of these of objects.
  - → Each object maintains a private copy of each attribute value
- Operation
  - An operation of a class is an identified common behavior of all instances of this class (i.e. set of concrete objects). The behavior operates on the class attributes and usually derives the change of a class' state.

7

Example: Class and instance



3. Message passing

- Message passing is the only way to interact with objects
  - Objects communicate by sending messages
  - Objects respond to a message that is sent to them by performing some action.

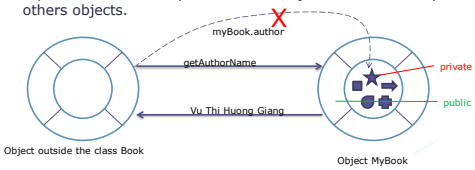
II. Encapsulation

- Encapsulation: Prevents the code and data being randomly accessed by other code defined outside the class.
  - Group data and operations performed on these data together in a class
  - Hide details about the class implementation from the user.

10

Visibility scope

- Scope determines the visibility of program elements with respect to other program elements.
- Given a class:
  - A private attribute or operation of an object is accessible by all others objects of the same class
  - A public attribute or operation of an object is accessible by all others objects.



If an attribute or operation is declared private, it cannot be accessed by anyone outside the class

11

III. CLASS BUILDING

1. Declaration
2. Data hiding

12

III.Class building
1. Declaration

a. Class declaration

```
[package package-name;]
[access-modifier] class class-name {
   // class body
   access-modifier type instance-variable-1; …
   static type class-variable-2; ...
   type method-name-1(parameter-list) {…} …
}
```

- A class encapsulates following members:
  - Name
  - Variables: declare the attributes of a class
  - Methods: declare the operations of class
- Related classes can be grouped into a package to easily control the access to these classes.
- The visibility scope of a class can be explicitly set by declaring an access modifier.

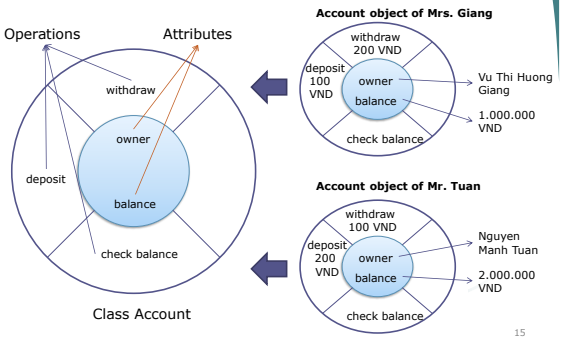
III.Class building
1. Declaration
a. Class declaration

Access modifier: Class' and members' visibility scope

- Four visibility scopes are identified in term of access modifiers to limit the access to the attribute and the operation:
  - public: visible to the world (free access)
  - private: visible to the class only (class access)
  - protected: visible to the package and subclasses (only with an inheritance relationship)
  - default (no modifier declared): visible to the package (package access)
- Example:

```
package acu.java.example;
class Account {
…
}
```


Example: Class and Object



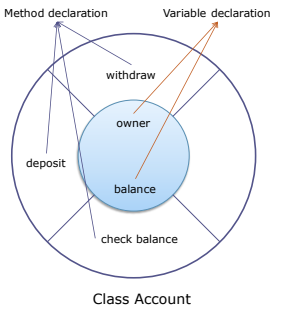
III. Class building
1. Declaration
a. Class declaration

b. Variable declaration

- Variables must be declared within a class
  - Instance variables (member variables): declared within a class + outside any method
  - Class variables : declared within a class, outside any method, with the static keyword
  - Local variables: declared inside methods, constructors or statement blocks (invisible outside them)

```
class class-name {
   [access-modifier] type instance-variable-1; …
   static datatype class-variable-1; ...
   datatype method-name-1(parameter-list) {
        datatype local-variable-1;
        …
   } …
}
```


Example: Class and variables



- Declaration of a class with two variable members:

```
class Account{
   String owner;
   long balance;
…
}

class Account1{
   private String owner;
   private long balance;
…
}
```

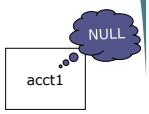
Can you show the difference between these 2 declarations ?
→ Use private modifier for declaring the concrete state of an object.

Variable creation

`datatype var1 = new datatype();`

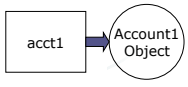
- A variable **`var1`** is declared to refer to the object of class **`datatype`**.

```
Account1 acct1;
// acct1 represents nothing
```

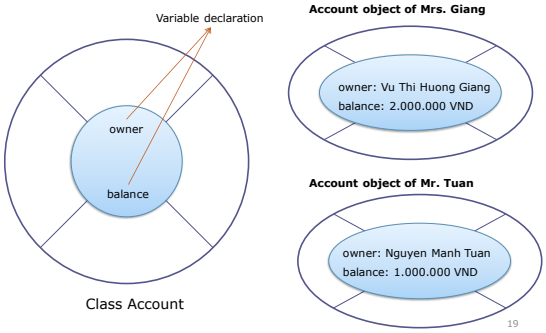


- After the assignment, the reference variable **`var1`** contains the address of a concrete **`datatype`** object that was created in memory.

```
acct1 = new Account1();
// acc1 refers to an Account1 object
```



Example: instance variable creation

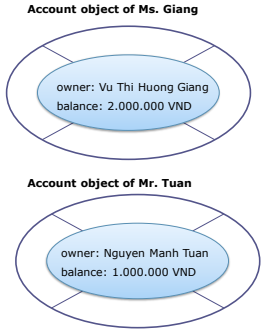


Example: instance variable creation

- New Account objects are created and new values are assigned to theirs 2 variables:

```
Account acc1 = new Account();
acc1.name =
   "Vu Thi Huong Giang";
acc1.balance = "2000000";
Account acc2 = new Account();
acc2.name = "Nguyen Manh Tuan";
acc2.balance = "1000000";
```

→ each object has its own data space
→ changing the value of a variable in one object doesn't change it for other



III. Class building
1. Declaration
a. Class declaration
b. Variable declaration

Class (static) variables

- Class variable: declared with the **`static`** keyword
- Memory space for a class variable is created when the class is first referenced
- It holds the same value for all objects instantiated from this class → changing the value of a static variable in one object changes it for all others
- Declaration format:

```
[access-modifier] static data-type
      member-variable-name;
```

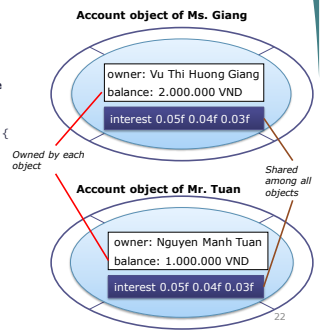
- Reference format:

```
class-name.member-variable-name;
object-name.member-variable-name;
```


Static variable

```
class Account {
// Member variable
String name; // Account name
long balance; // Balance
static float interest;//Deposit rate
}
class AccountClassUsage{
 public static void main(String[] args) {
      Account.interest=0.05f;
      Account giang= new Account();
      Acount tuan = new Account();
      giang.interest=0.04f;
      tuan.interest=0.03f;
 }
}
```

→ **Static variable cannot be used within a non-static method.**



III. Class building
1. Declaration
a. Class declaration
b. Variable declaration

c. Method declaration

- A Java method has two parts:
  - Declaration:
    - Specifies the name of the method, its return type and its formal parameters (if any)
    - Is used to hide the internal data structures of a class, as well as for their own internal use.
  - Implementation:
    - Specifies a collection of statements that are grouped together to perform an operation.
    - These statements are executed when a method is called.
    - It is through the implementation of the method that the state of an object is changed or accessed.

III. Class building
1. Declaration
a. Class declaration
b. Variable declaration

c. Method declaration

- Syntax

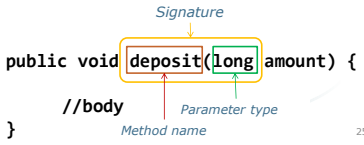
```
[access-modifier] return-type name(parameter-list) {
   // body
   [return return-value;]
}
```

where:
  - return-type: type of values returned by the method (void if a method does not return any value)
  - name: name of the method
  - parameter-list: sequence of type-identifier lists separated by commas
  - return-value: what value is returned by the method.

III. Class building
1. Declaration
a. Class declaration
b. Variable declaration
c. Method declaration

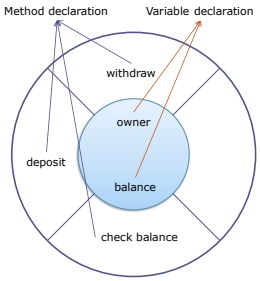
Signature

- The signature of a method consists of
  - The method's name
  - A list of parameter types, in which the number and the order of parameters are respected.
- Example: the signature of the following method is **deposit(long)**



25

Example: Method declaration and implementation



Class Account

- The type of a variable assigned the value returned by a method must agree with the return type of this method:

```
class Account1{
   private String name;
   private long balance;
   // deposit money
   public void deposit(long money) {
         balance += money;
   }
   // Check the account balance
   public long checkBalance() {
         return balance;
   }
...
}
```

- Member variables can be used anywhere inside the class.

26

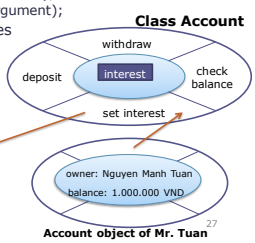
Static method

static return-type name(parameter-list) { … }

- Several restrictions:
  - can only call static methods
    - Class name.method_name(argument);
    - Object name.method_name(argument);
  - must only access static variables

```
class Account {
// Member variable
...
static float interest;//Deposit rate
public static void setInterest(float pInterest){
    interest = pInterest;
}
}

Account.setInterest(0.05f);
Account tuan = new Account();
tuan.setInterest(0.04f);
```



27

Method with a variable-length argument

- Declaring a variable-length parameter allows passing many values of the same data type.
  - No explicit number and order of arguments is declared
  - You may not pass the value to a variable-length argument.
- The passed arguments are treated as an array.
- Syntax:

```
modifier return-type method-name
   (data-type variable-length-parameter) {
   // body
   return return-value;
}
```

28

Example

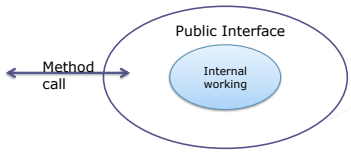
```
public class VariableLengthArgumentUsage {
    public static double average( double... numbers ) {
        double total = 0.0;
        for ( double d : numbers )
          total += d;
        return total / numbers.length;
    }
    public static void main(String[] args) {
        double d[] = {10.0, 20.0, 30.0, 40.0, 50.0};
        System.out.printf("Mean value of this array is");
        System.out.printf(" %.1f\n", average(d[0], d[1], d[2],
          d[3], d[4]));
    }
}
```

29

III. Class building
1. Declaration

2. Data hiding

- A class provides data hiding
  - Data is in a unique scope; access controlled with public, private, protected keywords
  - Data is accessed through public methods; a collection of the signatures of public methods of a class is called interface.
  - → Avoid illegal modification of attributes



30

Quiz 1 – variable and method declaration

- Declare the class Media describing media products such as book, that has following attributes and operations
  - Title
  - Category
  - Price
  - Show the title, price, category of a media product
  - Set a title, price, category for a media product

Quiz 1 – solution

```
public class Media {
    private String title;
    private String category;
    private float cost;
    public String getTitle() {
      return title;
    }
    public String getCategory() {
      return category;
    }
    public float getCost() {
      return cost;
    }
    public void setCategory
      (String category) {
      this.category = category;
    }
    public void setCost
      (float cost) {
      this.cost = cost;
    }
    public void setTitle
      (String title) {
      this.title = title;
    }
}
```


Quiz – static variable and static method

2. In the class Media:
   - Declare the product distributor as a static variable
   - Declare and implement the static method that allows changing the distributor of all media products.
3. In the class Media:
   - Declare and implement a method for displaying information about a concrete media product. Explain which attributes can be displayed by this method and why.
   - Declare and implement a method with variable-length argument for displaying all information about a concrete media product.
4. Write a program that
   - calls the static method using the class name
   - calls an instance of the method with/without variable-length argument

Quiz 2 - Solution

```
public class Media {
   private String title;
   private String category;
   private float cost;

   static String distributor;

   // ... implementation of methods as in the solution of Quiz 1

   public static void setDistributor(String dist) {
        distributor = dist;
   }
}
```


Quiz 3 - Solution

```
public void displayInstanceInfo() {
    System.out.println("Information about a
    concrete media product: ");
    System.out.println("Title: "+ title);
    System.out.println("Category:" +
    category);
    System.out.println("Price:" + cost);
    // can not access to static variable
}

public static void displayStaticInfo() {
    System.out.println("Distributor: " +
    distributor);
    // can not access to instance variable
}
```

```
public static void displayInstanceInfo
    (String... values) {

    System.out.println("Information
    about the media product: ");

    for ( String v : values )
          System.out.print(v);
}
```


Quiz 4 - solution

```
public class MediaClassUsage {
     public static void main(String[] args) {
          // static reference of variable
          System.out.println("Distributor: " + Media.distributor);
          // create a concrete media product
          Media media = new Media();
          media.setTitle("Vivaldi: The Four Seasons");
          media.setCategory("CD"); media.setCost((float) 14.38);
          media.setDistributor("Amazon");
          // static method call
          Media.displayStaticInfo();
          // call of instance method without variable-length args
          media.displayInstanceInfo();
          // call of instance method with variable-length args
          media.displayInstanceInfo("Title: ", media.getTitle(), "\n",
            "Category: ", media.getCategory(),"\n",
            "Price: ", Float.toString(media.getCost()),"\n",
            "Distributor: ", media.distributor);
     }
}
```

Review

- Data abstraction:
  - the class is the abstract data type, defining the representation of and operations on objects of the type.
- Encapsulation:
  - Class encapsulates objects
  - Object encapsulates data and implementation
- Visibility: To control the access to a class and to its members, we can use
  - Access modifiers: public, private, protected, default
    - Private: for hidden members
    - Public: for interface members
    - Protected: for inheritance (discuss later)
  - Package scope: members of all classes in a package that have default modifiers are visible throughout the package
- Class building
  - Variable declaration
  - Method declaration

37

Review

- Data abstraction:
  - the class is the abstract data type, defining the representation of and operations on objects of the type.
- Encapsulation:
  - Class encapsulates objects
  - Object encapsulates data and implementation

38

Review

- Visibility: To control the access to a class and to its members, we can use
  - Access modifiers: public, private, protected, default
    - Private: for hidden members
    - Public: for interface members
    - Protected: for inheritance (discuss later)
  - Package scope: members of all classes in a package that have default modifiers are visible throughout the package
- Class building
  - Variable declaration
  - Method declaration

39