LESSON III
Class Building

Trinh Thanh TRUNG (MSc)
trungtt@soict.hust.edu.vn
094.666.8608

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
  - Visibility
- Class building
  - Declaration
    - Class declaration
    - Class member declaration
  - Data hiding

Abstraction

- "Abstraction – a concept or idea not associated with any specific instance"
  - E.g. Mathematics definitions
- Two types of abstraction
  - Control abstraction
    - Using subprogram and control flow
    - E.g. a:=(1+2)*5
  - Data abstraction
    - E.g. data type

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.

Abstraction

- Similarities
  - Mobile phone
  - Candybar style
- Differences
  - Business phone, music phone, 3G phone…
  - QWERTY keyboard, basic keys, touch control…
  - Colours, sizes…

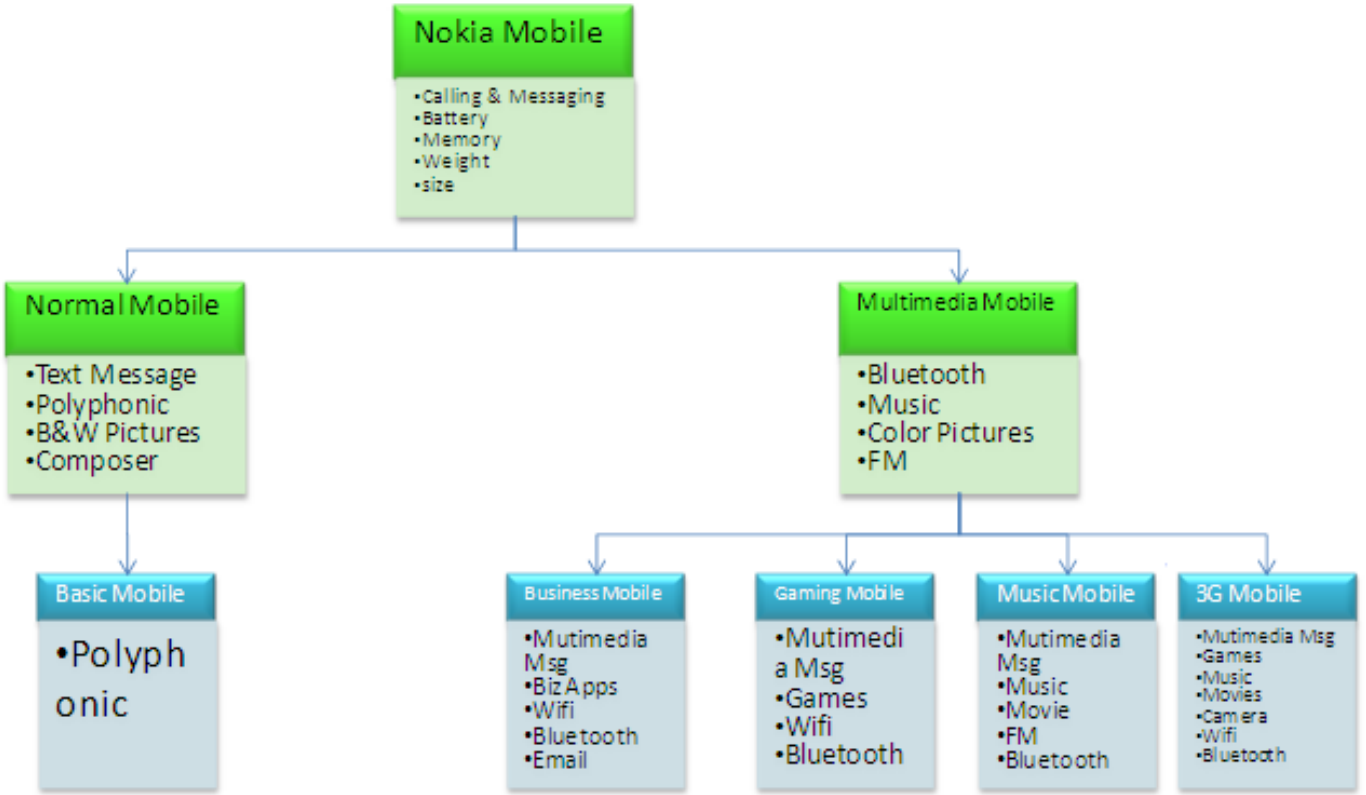
Abstraction

- Classify into different categories

Abstraction

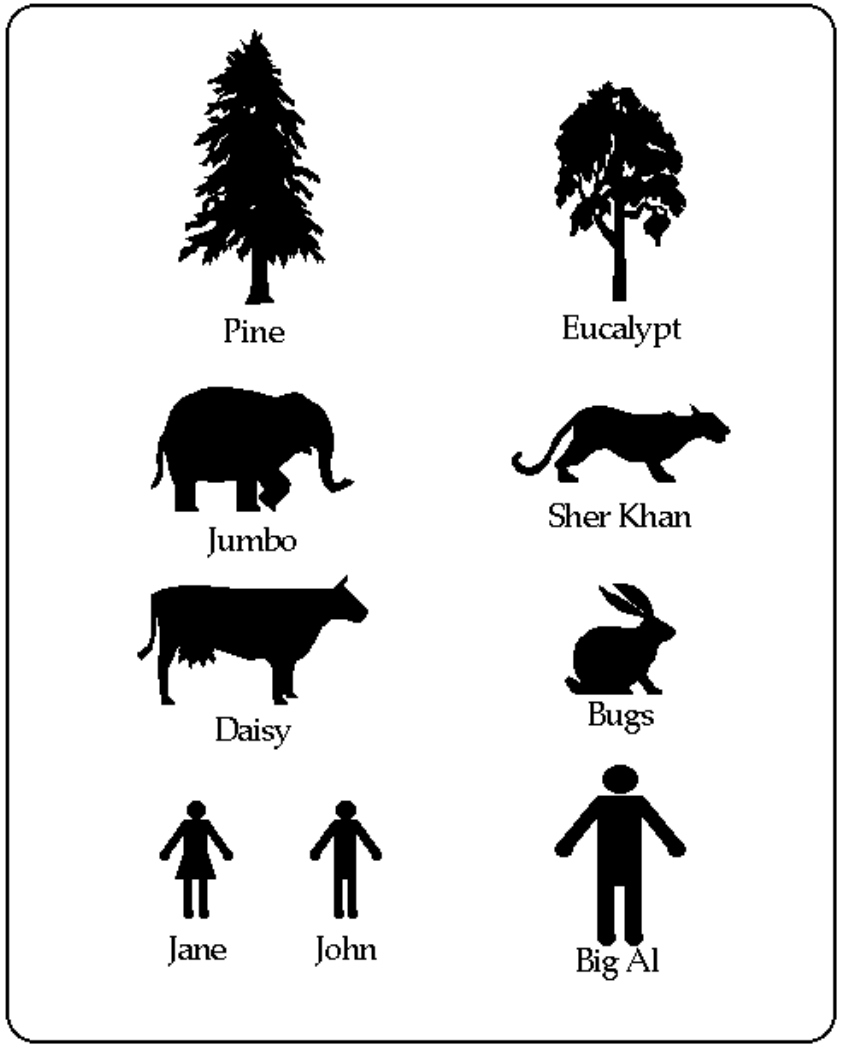
I. Data abstraction
1. Overview

- Group similarity specifications



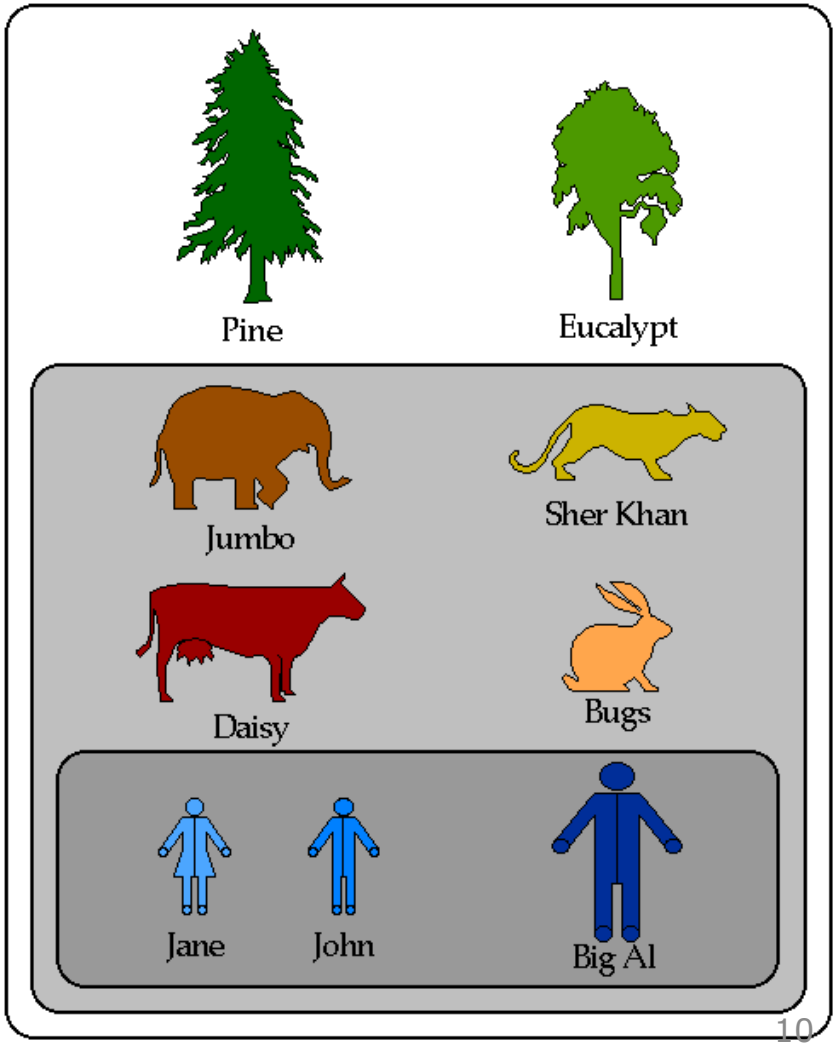
Abstraction

- unspecified objects



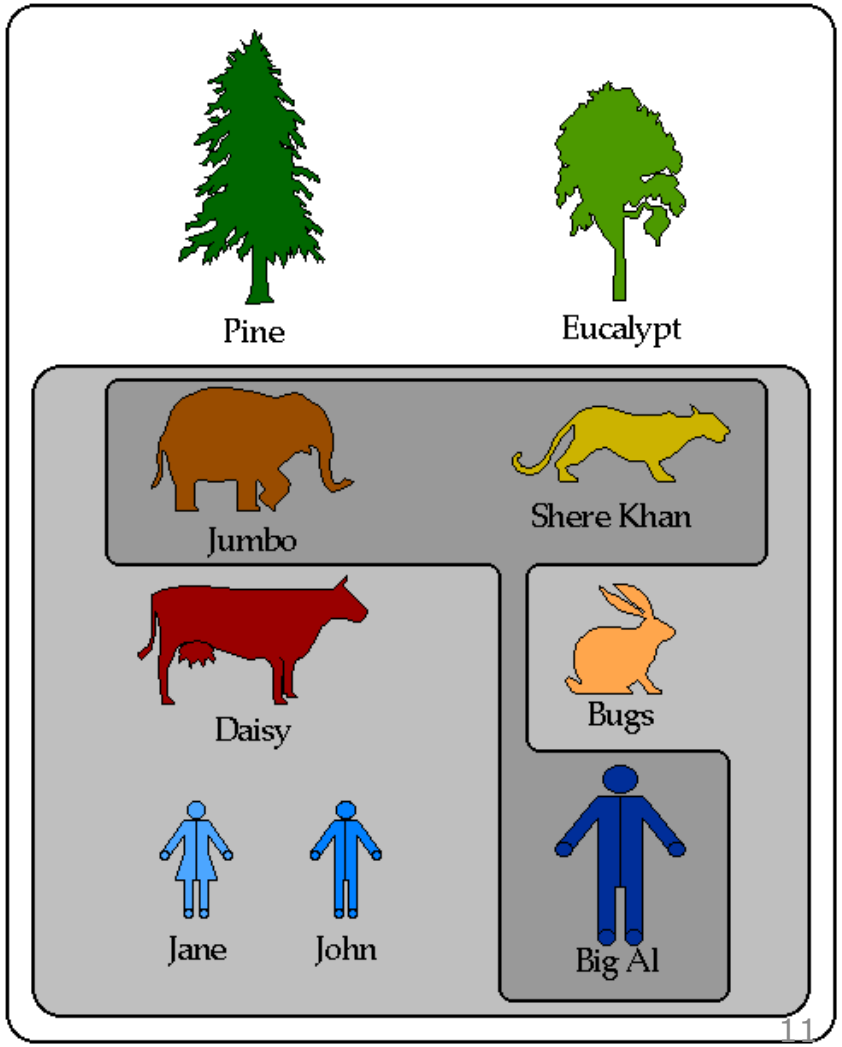
Abstraction

- organisms, mammals, humans



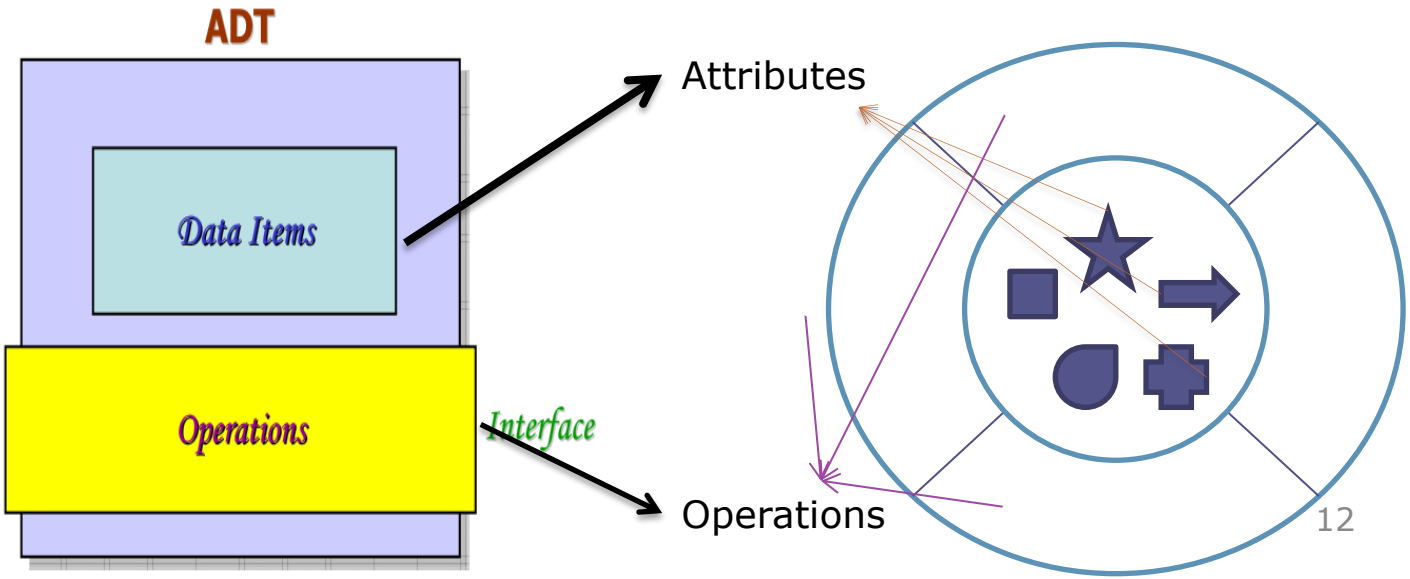
Abstraction

- organisms,
  mammals,
  dangerous
  mammals



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

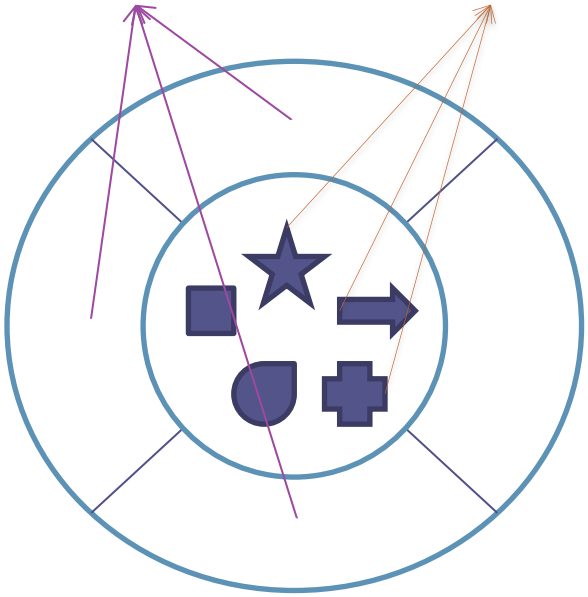
- Attribute
  - An attribute of a class is an identified common state of all instances of this class (i.e. set of concrete objects).
  - An attribute specifies all possible values that can be assigned as concrete state of these of objects.

  → 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.

Example: Class and instance

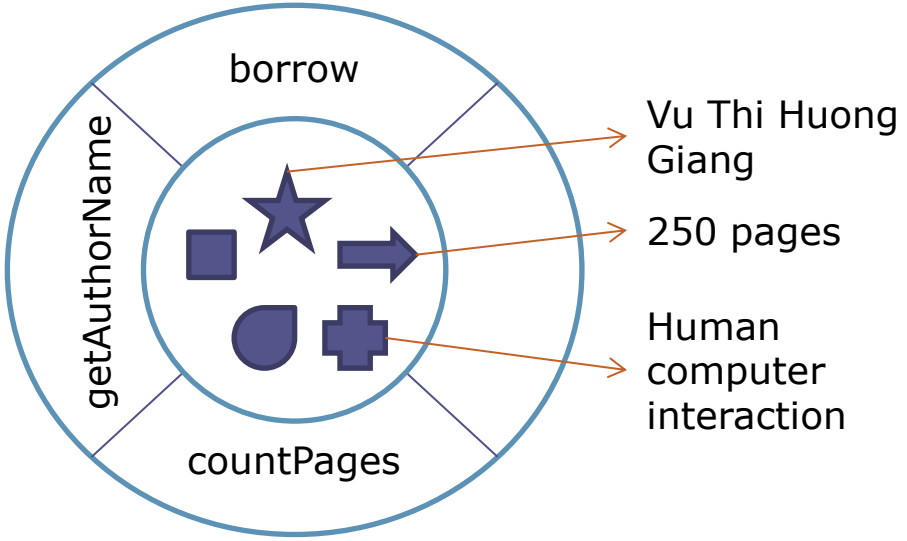
Operations: object behavior

Attributes: information about object states



Class

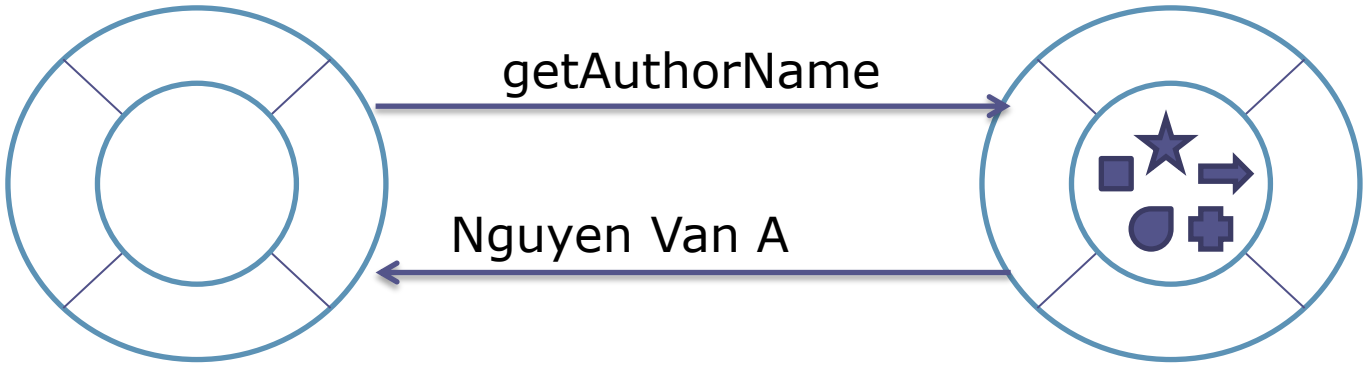
Instance: a concrete object
Instance attribute: attribute that is assigned by a concrete value



Object MyBook

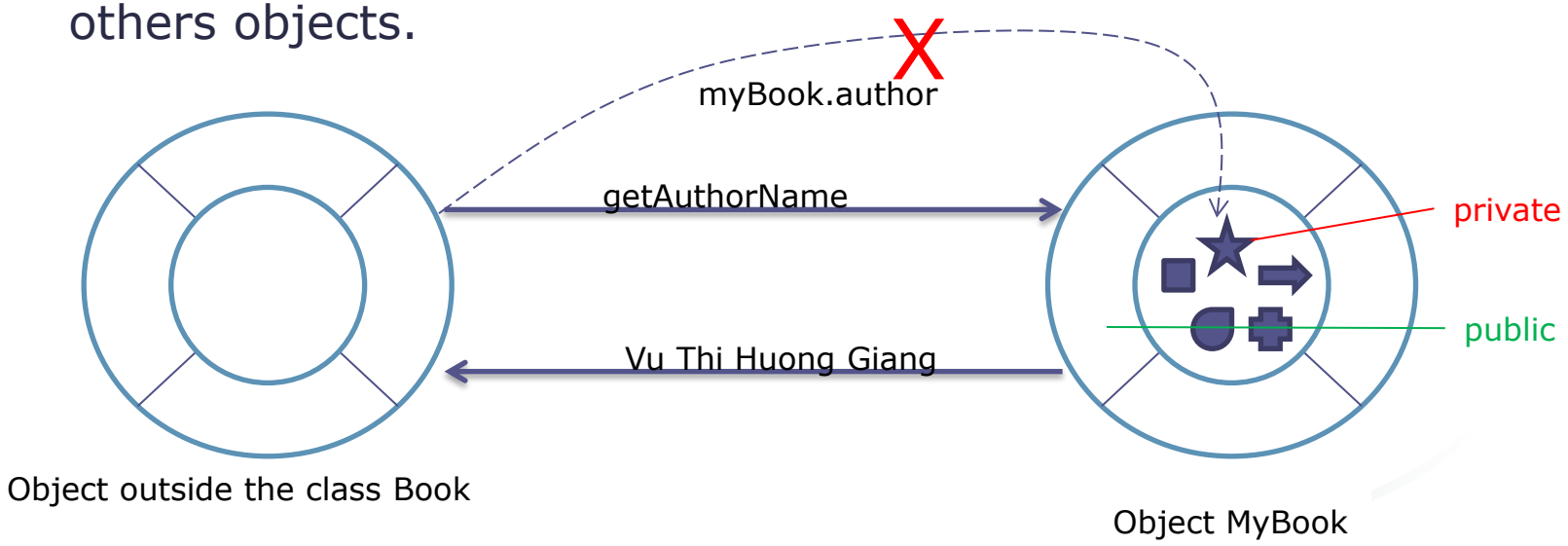
3. Message passing

- 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.



4. 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

III. CLASS BUILDING

1. Package
2. Declaration
3. Data hiding

Package

- A *package* is a like a folder
  - grouping of related types providing access protection and name space management
  - Avoid name conflict
  - Protect data
- Existing Java packages
  - java.lang
  - javax.swing
  - java.io
  - …

Package

- Syntax

  `package <package name>;`

  – Must be the first line in the source file.

  – There can be only one package statement in each source file, and it applies to all types in the file.

  – *package name* is user-defined

    - E.g. `package ltu11b;`

  – A package can be put inside another package

    - Separated using '.'
    - E.g. `package oop.sie.ltu11b;`

Package

- Naming Conventions
  - Package names are written in all lower case to avoid conflict with the names of classes or interfaces.
  - Companies use their reversed Internet domain name to begin their package names
    - e.g. com.example.mypackage for a package by a programmer at example.com.

a. Class declaration

- Syntax

```
class class-name {
}
```

- Full syntax

```
[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) {…} …
}
```

- Example:

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

a. Class declaration

- 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.

Access modifier: Class' and members' visibility scope

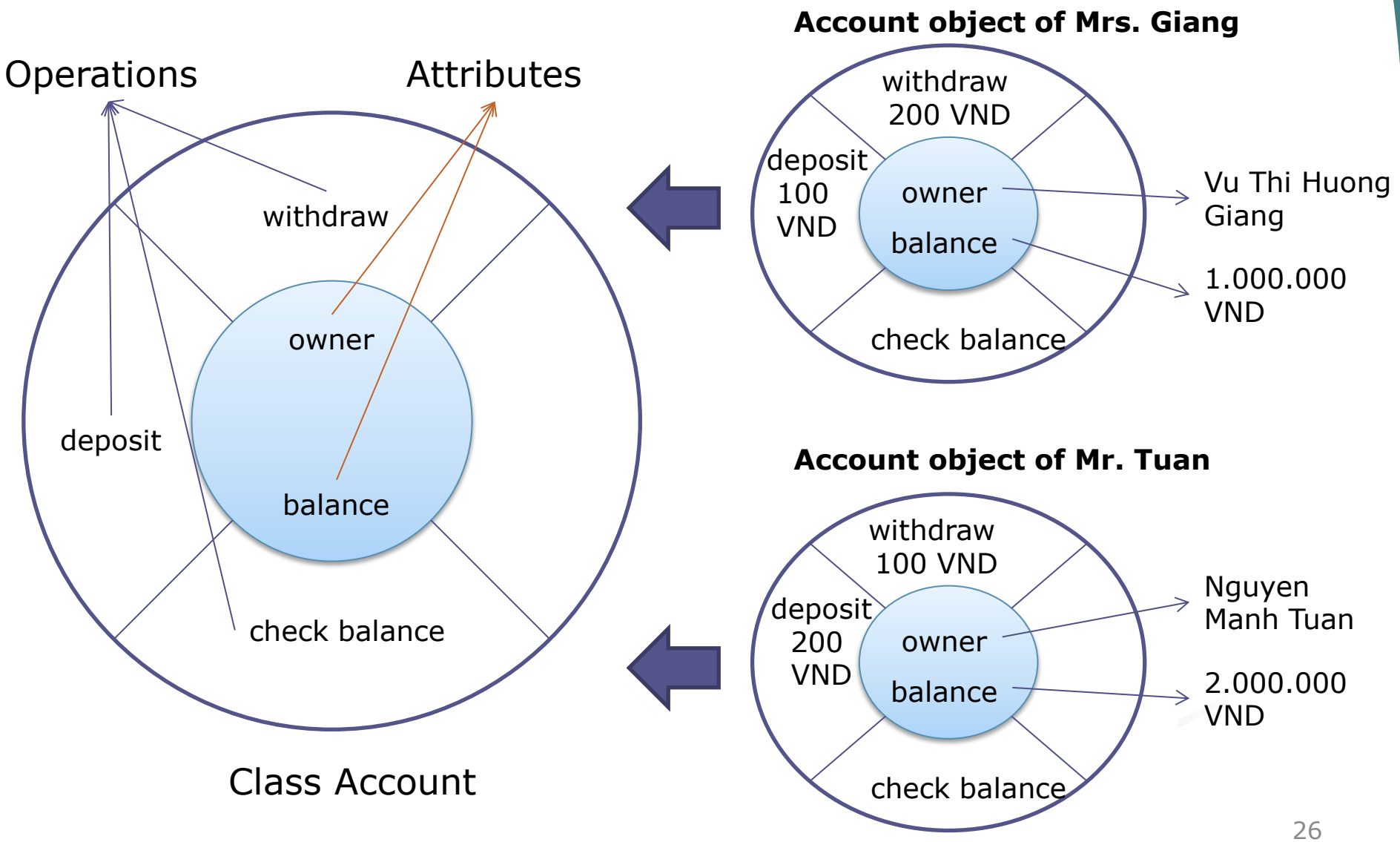
- Four visibility scopes are identified in term of access modifiers to limit the access to the attribute and the operation:
  - public: visible to the world (free access)
  - private: visible to the class only (class access)
  - protected: visible to the package and subclasses (only with an inheritance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

Access modifier: Class' and members' visibility scope

inner class

	public	default	private
Same class	Yes	Yes	Yes
Same package	Yes	Yes	No
Different package	Yes	No	No

Example: Class and Object

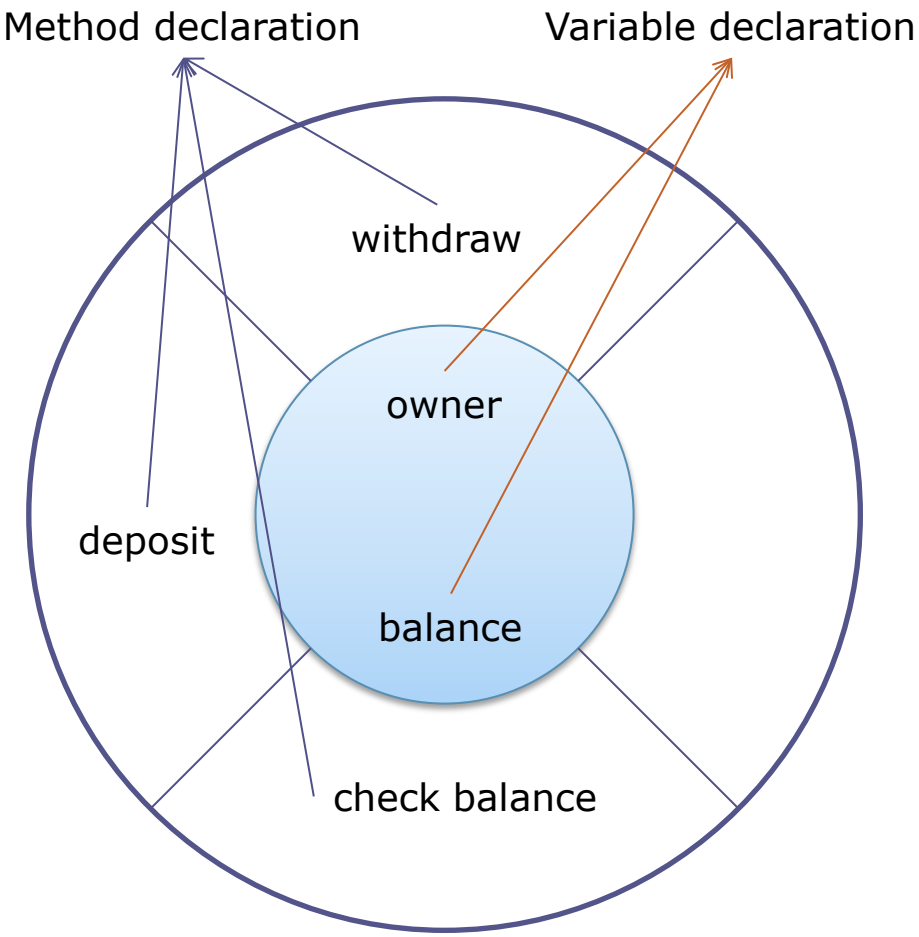


b. Variable declaration

- 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



Class Account

- 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 ?

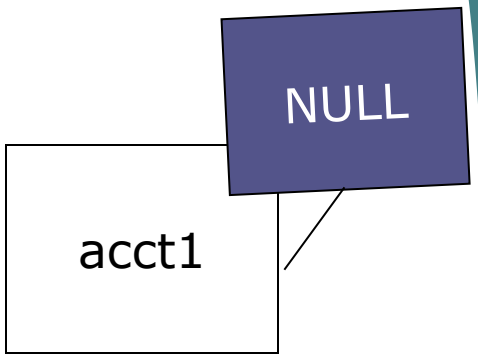
→ Use private modifier for declaring the concrete state of an object.

Variable creation

`datatype var1 = new datatype();`

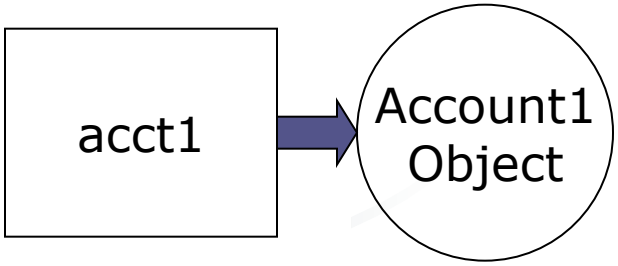
- A variable **`var1`** is declared to refer to the object of class **`datatype`**.

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

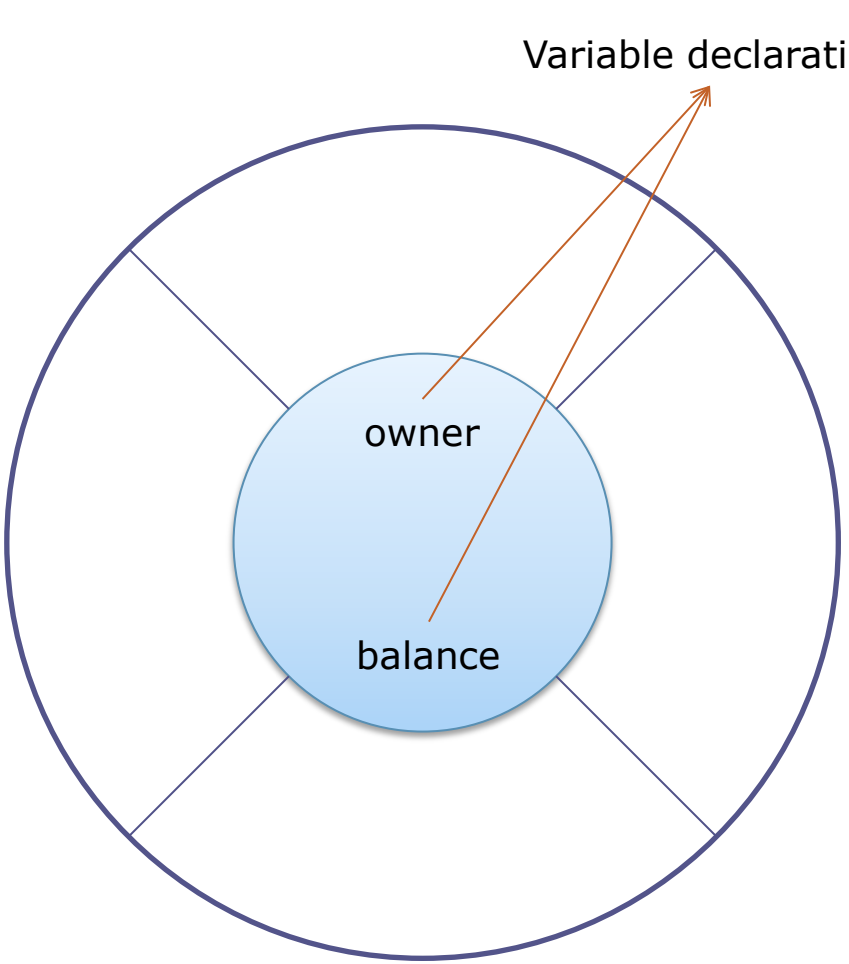


- After the assignment, the reference variable **`var1`** contains the address of a concrete **`datatype`** object that was created in memory.

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



Example: instance variable creation



Account object of Mrs. Giang

Account object of Mr. Tuan

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";
```

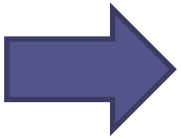
→ each object has its own data space

→ changing the value of a variable in one object doesn't change it for other

Account object of Ms. Giang

Account object of Mr. Tuan

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.

c. Method declaration

- Syntax

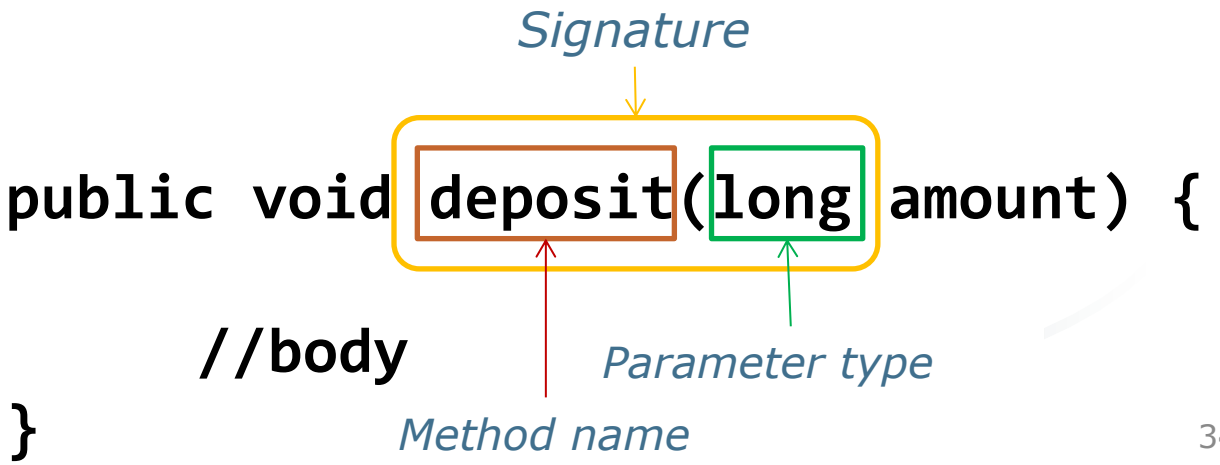
```
[access-modifier] return-type name(parameter-list) {
   // body
   [return return-value;]
}
```

where:

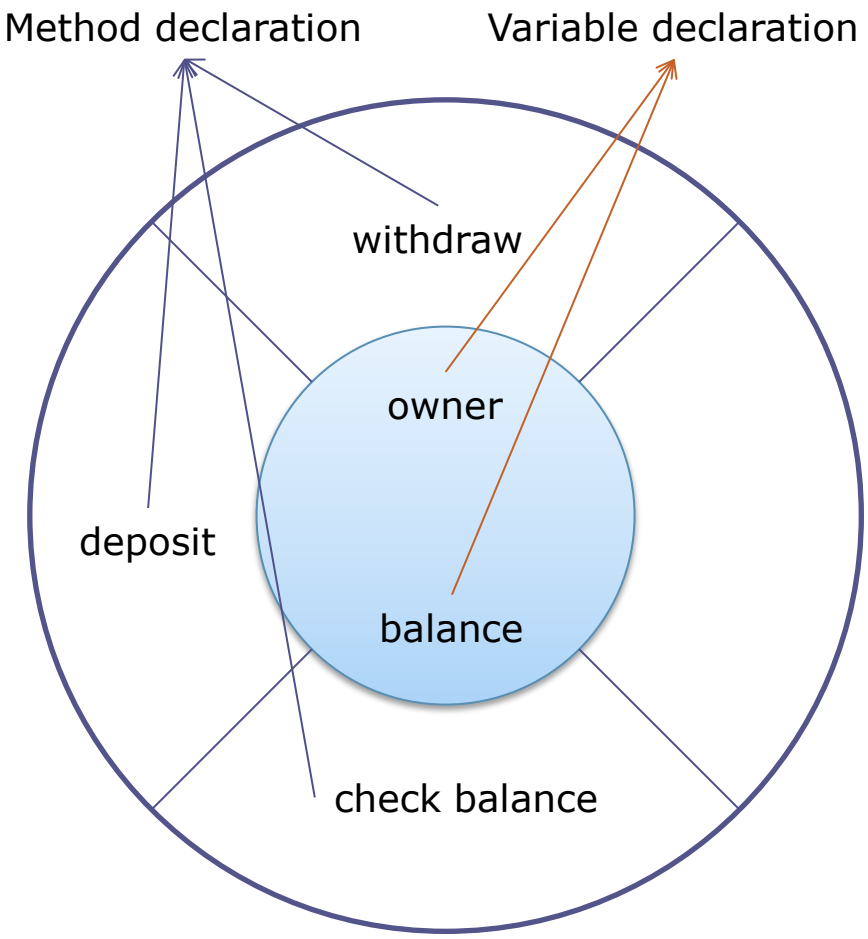
- return-type: type of values returned by the method (void if a method does not return any value)
- name: name of the method
- parameter-list: sequence of type-identifier lists separated by commas
- return-value: what value is returned by the method.

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)`**



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.

Review

- Data abstraction:
  - the class is the abstract data type, defining the representation of and operations on objects of the type.
- 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

Review

- Data abstraction:
  - the class is the abstract data type, defining the representation of and operations on objects of the type.

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