



OBJECT-ORIENTED PROGRAMMING LANGUAGES

Nguyen Hua Phung, Ph.D.
Department of CSE,
Uni. of Technology
2007

A decorative graphic at the top of the slide consists of two groups of three circles. The left group has a solid light purple circle on the left, a white circle with a light purple outline in the middle, and a white circle with a light purple outline on the right. The right group has a solid light purple circle on the left, a white circle with a light purple outline in the middle, and a solid light purple circle on the right.

Outline

- History
- Motivation
- OO Concepts
- Case study: Java

History



- 1960s Simula-67
 - objects
- 1970s Smalltalk
 - fully dynamic system, garbage collection
- mid-1980s C++
 - extension of C, graphical user interfaces
- mid-1990s Java
 - platform independence, virtual machine
- Recently C# VB.NET

Motivation



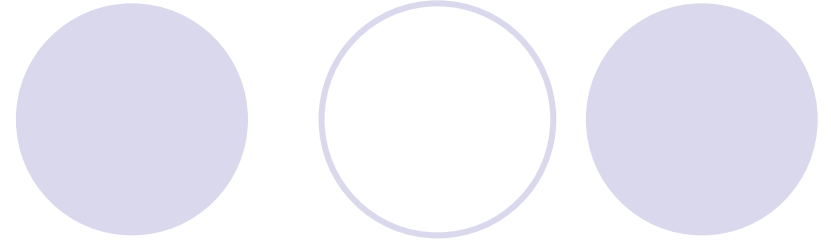
- **Software Reuse:**

- Extension of the data and/or operations
- Restriction of the data and/or operations
- Redefinition of one or more of the operations
- Abstraction
- Polymorphization

- **Software Independence**

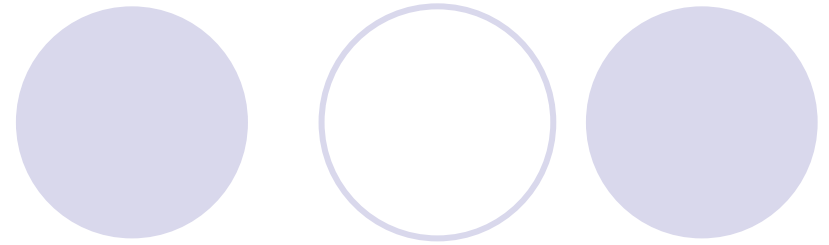
- Maintain the independence of different components
- Restricting access to internal details of software components

OO Concepts



- Class
- Object
- Method
- Message
- Inheritance
 - Single-inheritance
 - Multi-inheritance
- Encapsulation – Information Hiding
- Polymorphism

Class vs. Object



- A class defines the abstract characteristics of a thing
 - its **attributes** or **properties** and
 - its **behaviors** or methods or **features**.
 - A *Dog* has **fur** and is able to **bark**
- An object is a particular instance of a class.
 - Lassie is a dog

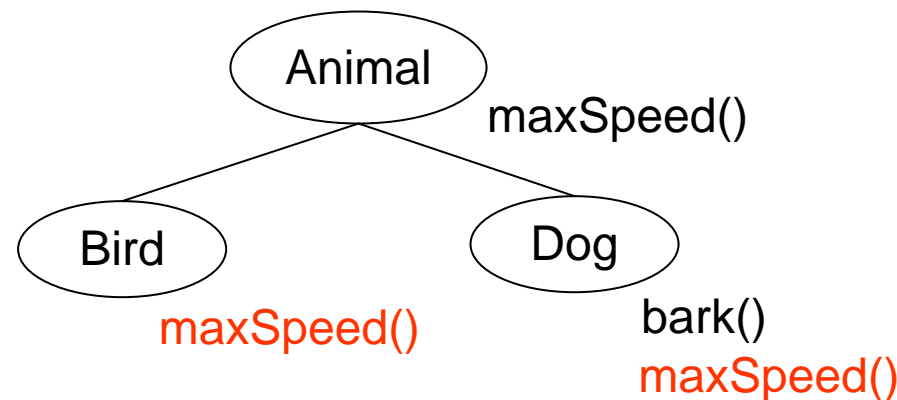
Method vs. Message



- A method describes a behavior of an object.
 - A dog can bark
- A message is a process at which a method of an object is invoked.
 - Lassie barks

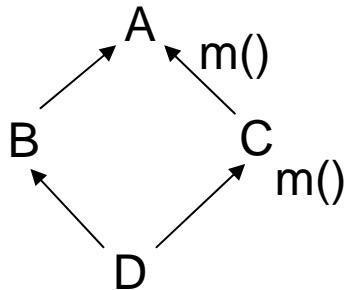
Inheritance

- Inheritance by B from A is a capability by which an object of B may have properties and methods defined in A as if they has been defined in B.



Single vs. Multiple Inheritance

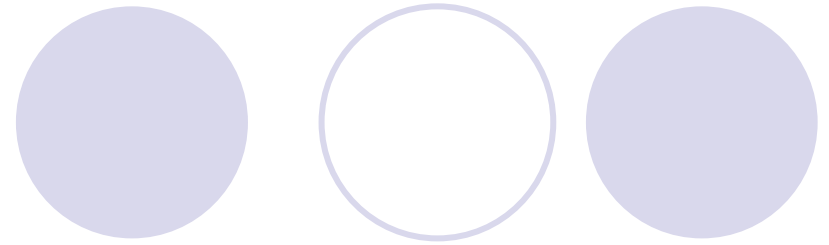
- Single inheritance: each class, except a root class, inherits from only one superclass
- Multiple inheritance: a class may inherit from two or more superclasses.



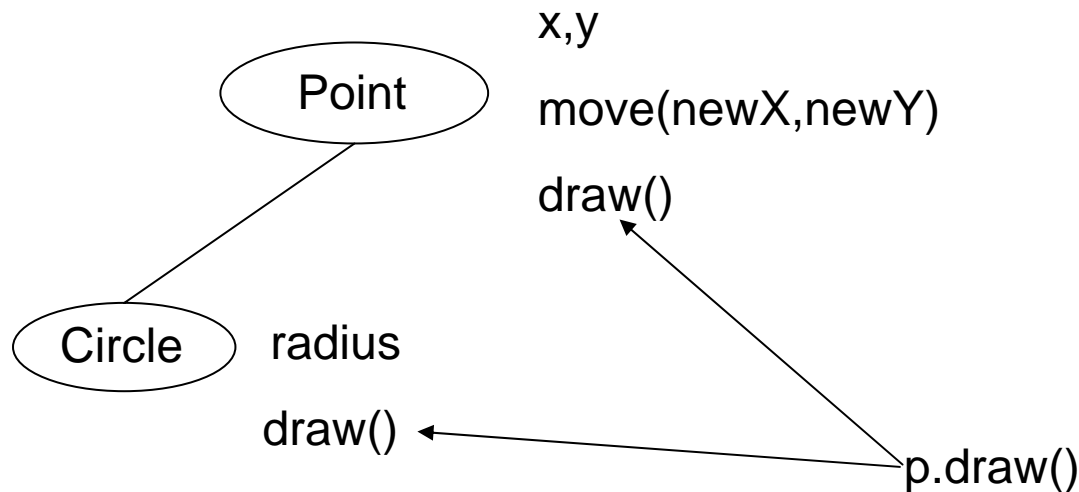
Encapsulation vs. Information Hiding

- Encapsulation: the grouping of related concepts into one item, such as a class or a component.
- Information hiding: the restriction of external access to attributes.

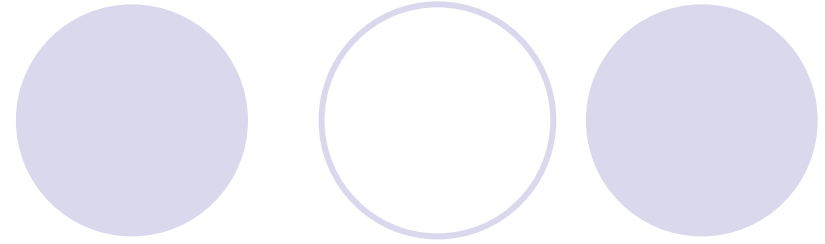
Polymorphism



- Polymorphism: different objects can respond to the same message in different ways.



Java programs

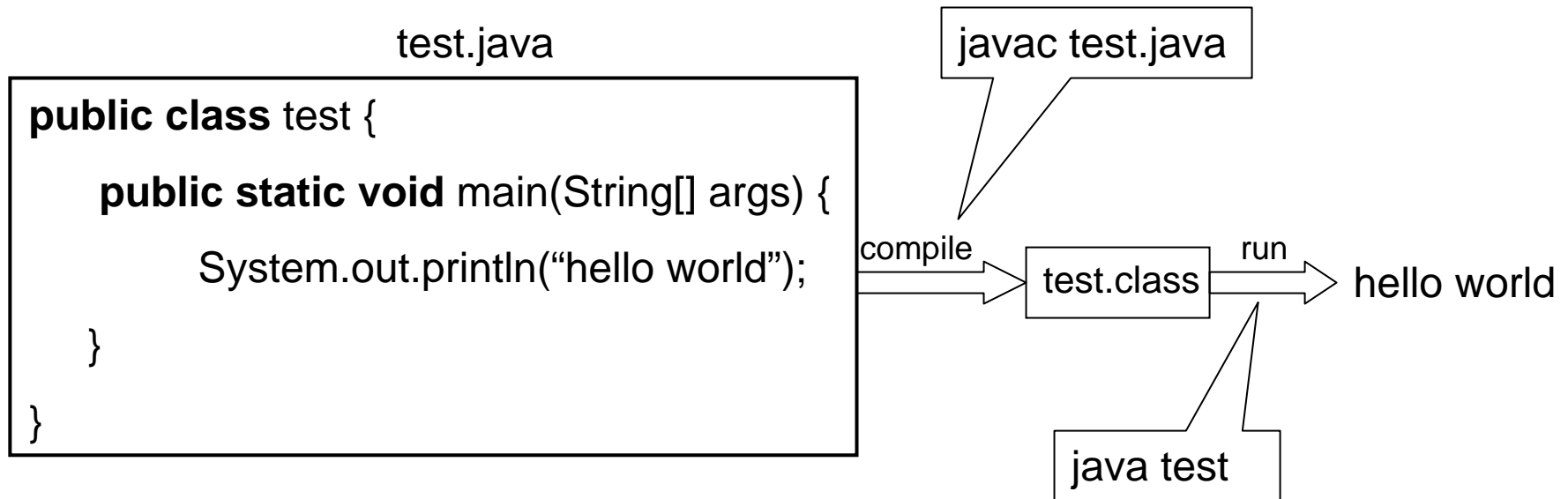


- Applications
- Applets

Java Application

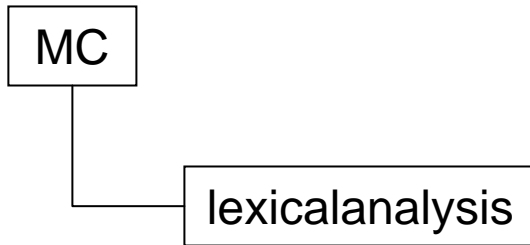
```
public static void main(String[] args) {...}
```

⇒ entry point



Package

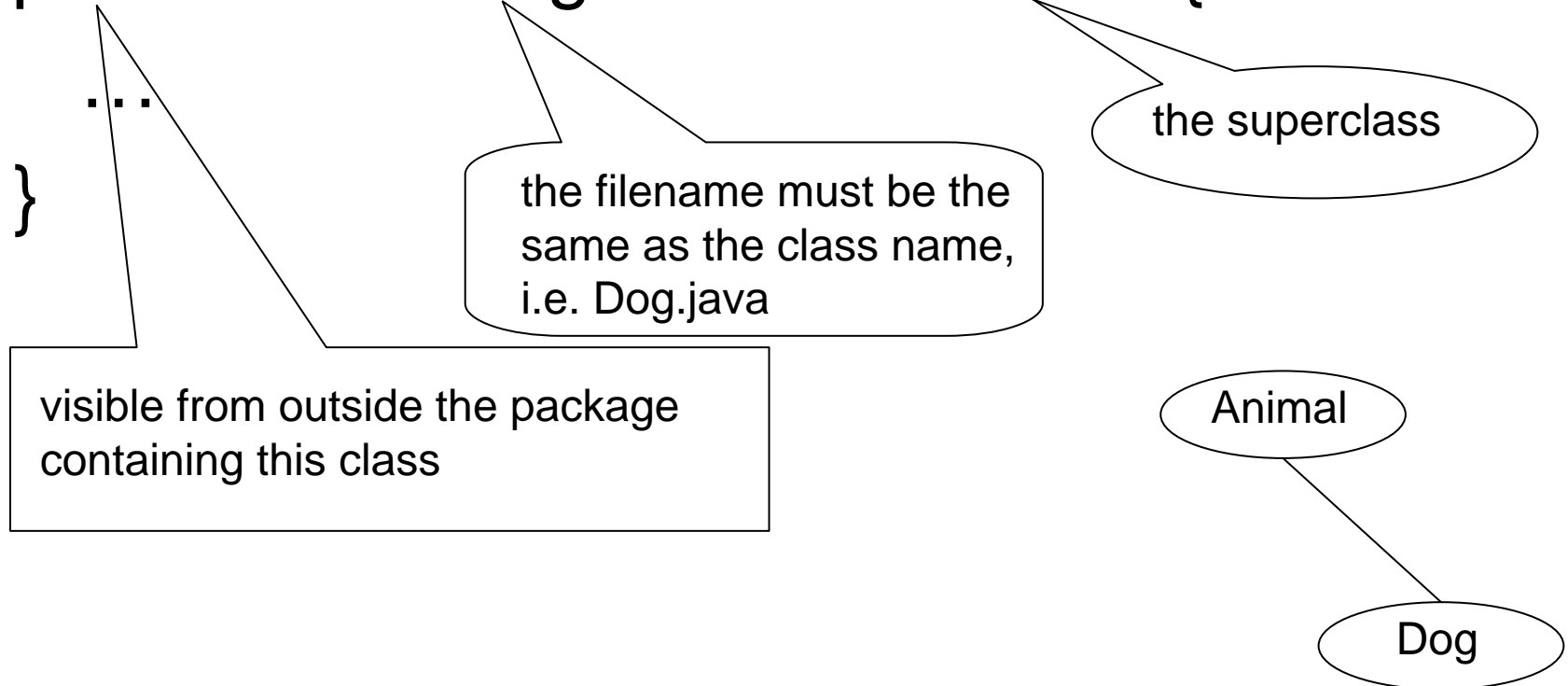
- package MC.lexicalanalysis



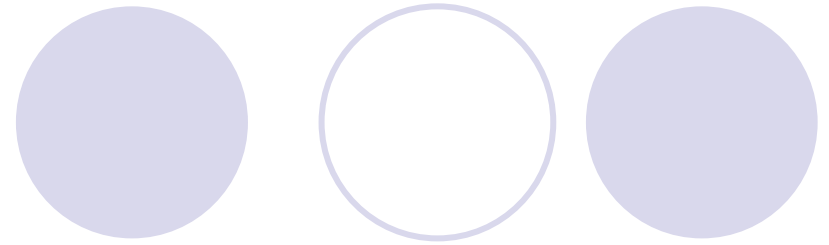
- divide a program into many groups
- enhance security

Java class

```
public class Dog extends Animal {
```

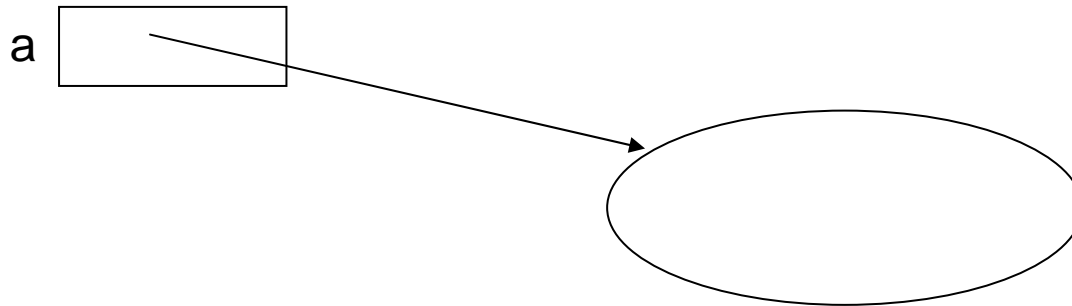


Object Handle



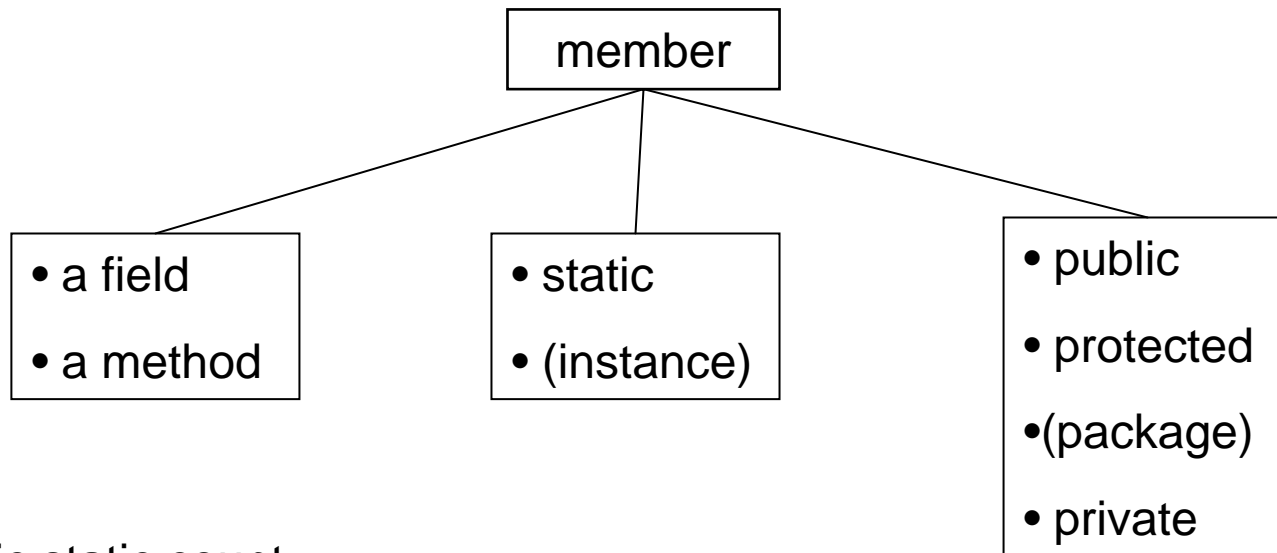
- Object handle is an identifier attached to an object when it is created.
- In Java, object handle is a virtual address.

`Dog a = new Dog();`



Java has no pointer type but a variable containing an object is really a pointer

Member of a class



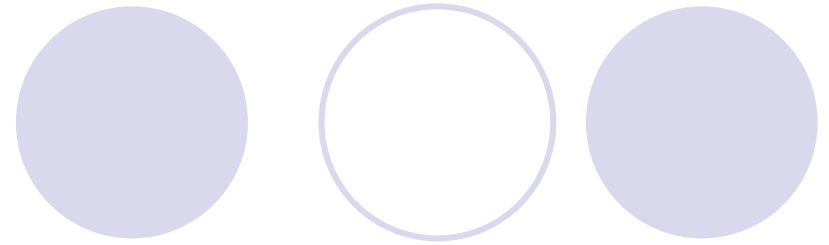
public static count;
protected void foo() {...}

Example



```
public class A {  
    public static int x,y;  
    private int z;  
    protected float t;  
    public static void foo() {...}  
    void goo(int x) {...}  
}  
class B extends A {  
    void goo(int x){...}  
}
```

Type checking



- Type equivalence

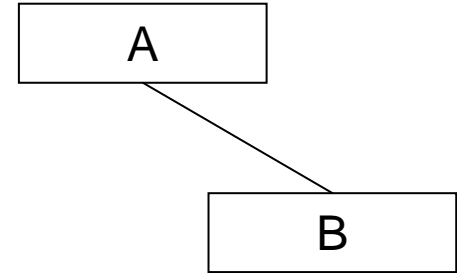
```
A x = new A();
```

```
B y = new B();
```

```
x = y;
```

```
y = x;
```

which one is right? why?



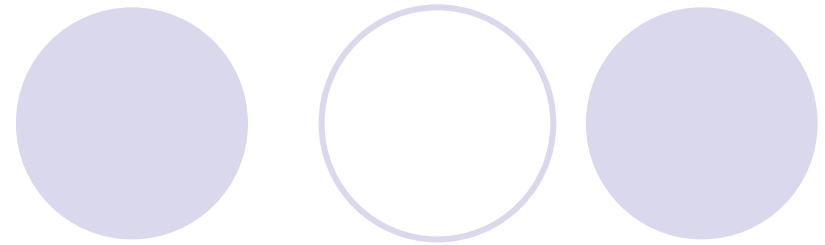
- Type casting

```
x = (A) y;
```

```
y = (B) x;
```

which one is necessary? why?

Polymorphism

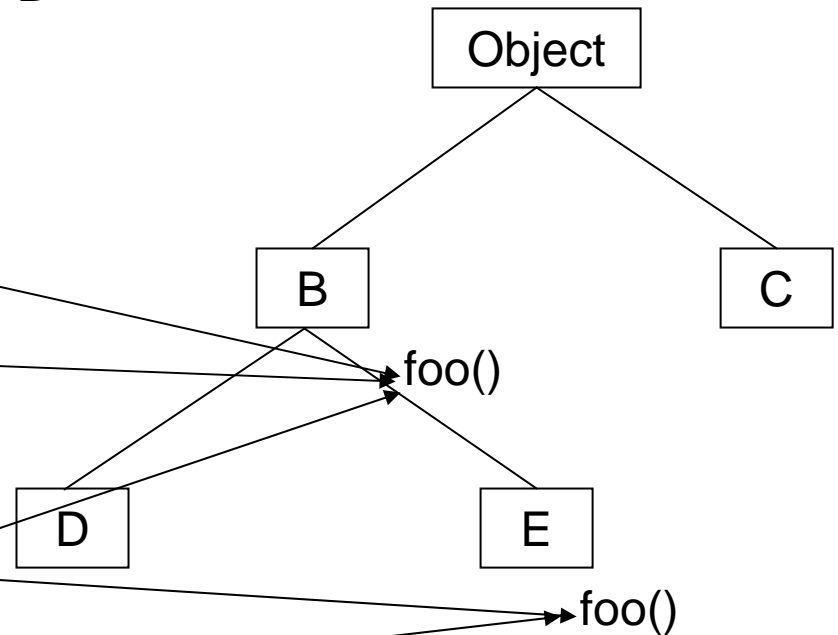


```
B x; // declare a variable of type B
x = new B(); // create an object of B
x.foo(); // send message foo to x

D y;
x = new D();
y.foo();

E z;
z = new E();
z.foo();

B t;
...
t.foo();
```



Useful links



- <http://java.sun.com/docs/books/tutorial/>
- <http://www.apl.jhu.edu/~hall/java/>