Object Oriented Approach

Polymorphism

- Polymorphism represents the capacity of an operation to be applied to objects of different classes
  - Redefinition of the code by keeping the same name of operation
  - The same ***operation*** can behave differently in several classes
  - The various methods which implement this same operation in the various classes must carry the same ***signature*** (name, parameters)
  - ***The implementation*** of the operation is determined by the class of the object to which it is applied

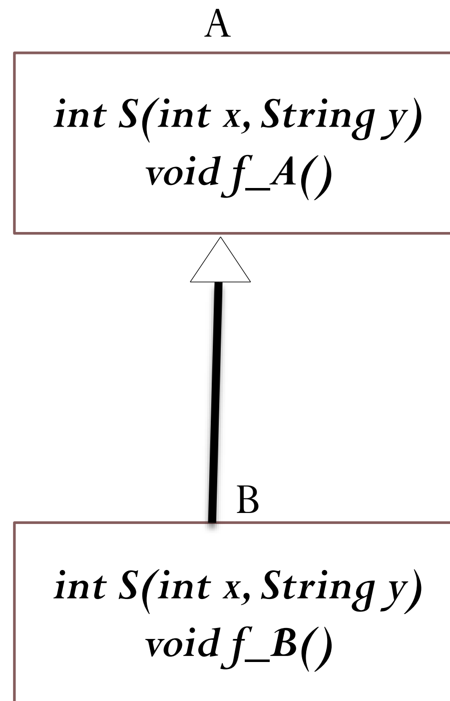
Polymorphism =
the same name, several implementations

Polymorphism

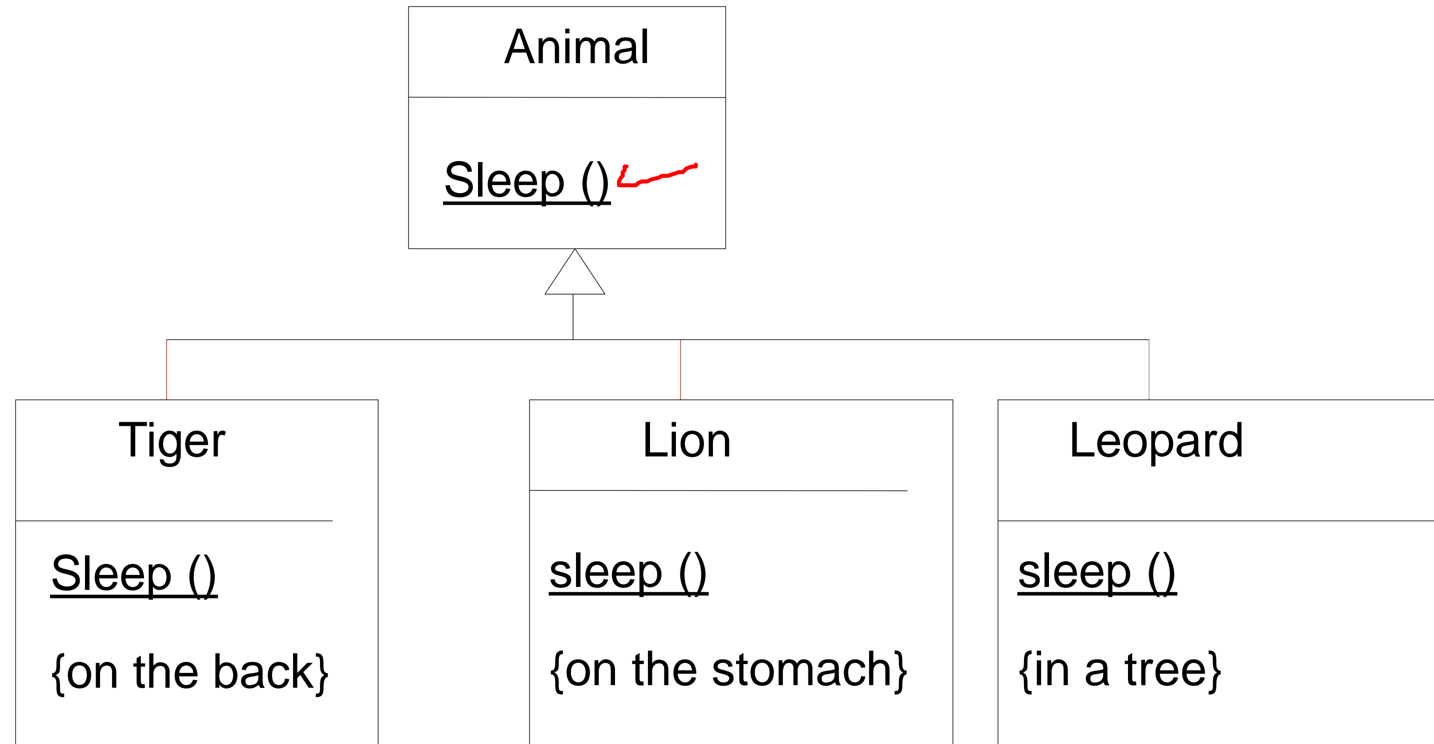
A a =new B();

b.S(5, “Msg");

B b=new A();//wrong



Polymorphism



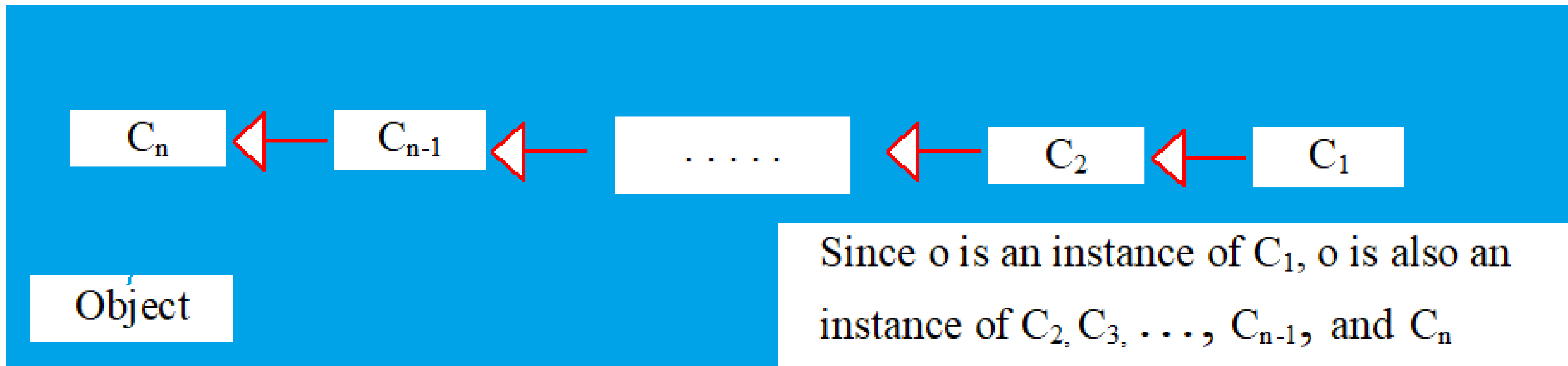
Polymorphism

- Can treat an object of a subclass as an object of its superclass.
- Suppose that the class Student inherits of the class Person.
  - Person p =new student();
  - Student s = new Peron(); ***Wrong***

- In Java, polymorphism refers to the dynamic binding mechanism that determines which method definition will be used when a method have been overridden.

- Method to be executed is determined at execution time, not compile time.

Dynamic binding



- If ***o*** invokes a method p, the JVM searches the implementation for the method ***p*** in $C_1, C_2, \ldots, C_{n-1}$ and C_n, in this order, until it is found. Once an implementation is found, the search stops and the first-found implementation is invoked.

Overriding methods in the Superclass

- A subclass inherits methods from a superclass.
- Sometimes it is necessary for the subclass to modify the implementation of a method defined in the superclass.
- This is referred to as *method overriding*.

Overloading

- **Method Overloading** is a feature that allows a class to have more than one **method** having the same name, if their argument lists are different.

- It is similar to constructor **overloading** in **Java**, that allows a class to have more than one constructor having different argument lists.

The instanceof operator

- The **java instanceof operator** is used to test whether the object is an instance of the specified type
- It returns either true or false
- An object of subclass type is also a type of parent class. For example, if Dog extends Animal then object of Dog can be referred by either Dog or Animal class.

The instanceof operator

```
class Animal{}
class Dog extends Animal{//Dog inherits Animal
  public static void main(String args[]){
     Dog d=new Dog();
    System.out.println(d instanceof Animal);//true
  }
}
```

Downcasting with java instanceof operator

- Dog d=**new** Animal();//Compilation error
- Dog d=(Dog)**new** Animal(); //Compiles successfully but ClassCastException is thrown at runtime
- **void** method(Animal a) {

```
if(a instanceof Dog){
  Dog d=(Dog)a;//downcasting
  System.out.println("ok downcasting performed");
}
```

Downcasting without the use of instanceof

```
static void method(Animal a) {
    Dog d = (Dog)a;//downcasting
    System.out.println("ok downcasting performed");
}
 public static void main (String [] args) {
  Animal a=new Dog();
  Dog.method(a);
}
```

Polymorphism

A a =new B();

b.S(5, “Msg");

a.f_B();//wrong

B b=(B) a;

b.f_B();

A a=new A();

B b=(B) a;//compile time ok

//runtime error