IS 2104 - Rapid Application Development

Object Oriented Concepts

Polymorphism

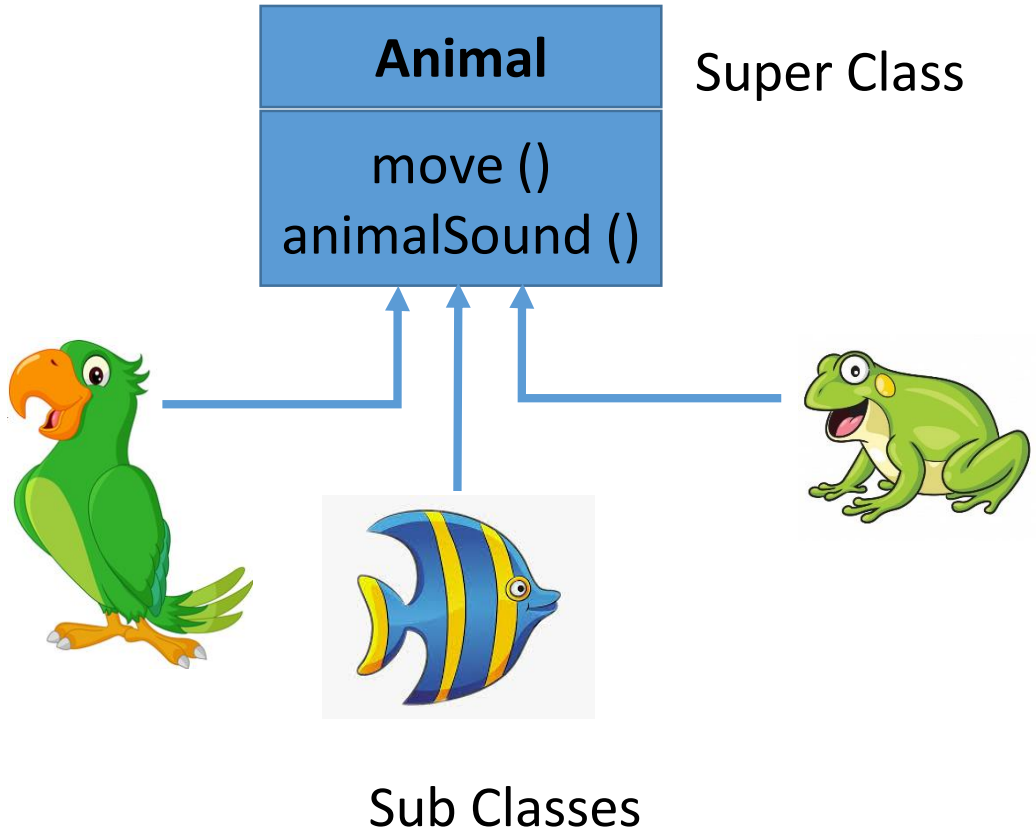
M.V.P. Thilini Lakshika
University of Colombo School of Computing
tlv@ucsc.cmb.ac.lk

Lesson Outline

- What is Polymorphism?
- Method Overloading
- Method Overriding
- Types of Polymorphism

Polymorphism

- Consider a program to simulates the movement and sound of several types of animals.
  Ex: Classes for Bird, Fish and Frog
- Each class extends superclass Animal, which contains methods move and animalSound.
- But, each specific type of animal responds to move message and animalSound message in its own way.
- Each object need to do what is appropriate for that type of object in response to the same method call.
- So, each subclass should implements method move and animalSound accordingly.



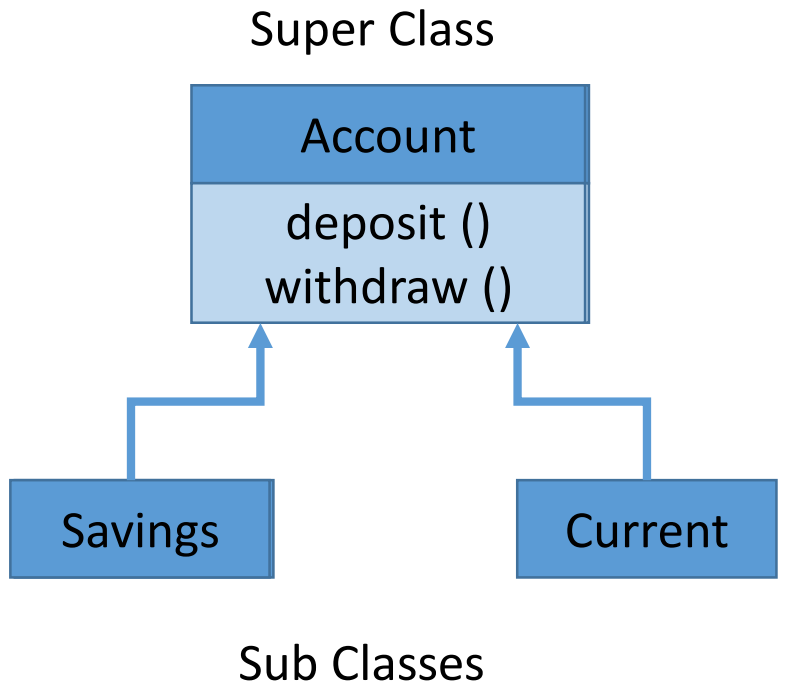
Polymorphism

- According to the above example, the same message (move and animal Sound) sent to a variety of objects has many forms.
- Polymorphism is the ability of an object to take on many forms.
- You can use polymorphism when a parent class reference is used to refer to a child class object.
- Polymorphism enables you to program in the general rather than program in the specific.
- Polymorphism allows to design and implement systems that are easily extensible.
  - New classes can be added with little or no modification to the general portions of the program, as long as new classes are part of the inheritance hierarchy.

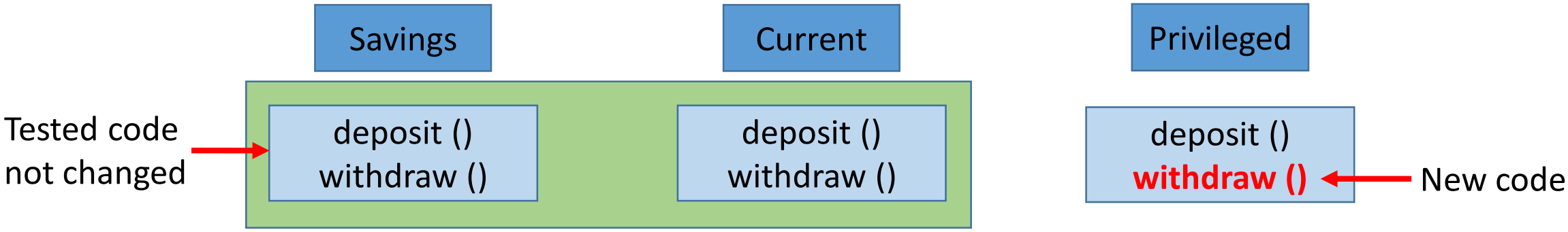
Polymorphism

- The operation of deposit and withdraw is same for Saving and Current accounts.
- So the inherited methods from Account class will work for both sub classes.
- You are supposed to add new class Privileged Banking Account with Overdraft Facility.

  Overdraft facility allows to withdraw an amount more than the available balance in your account.
- So, withdraw method for privileged account needs to implement newly.
- You can do this without changing the tested piece of code in Savings and Checking account.



Polymorphism



- When the withdrawn method for the saving account is called, a method from parent account class is executed.
- When the withdraw method for the privileged account is called, withdraw method defined in the privileged class is executed.
- This is **Polymorphism.**

Method Overloading

- The methods with same name, but with different signatures (parameter list) exist in the same class.

  Ex: void sum (int a, int b);

  void sum (int a, int b, int c);

  void sum (float a, double b);

```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
    a.p(10.0);
  }
}

class B {
  public void p(double i) {
    System.out.println(i * 2);
  }
}

class A extends B {
  // This method overloads the method in B
  public void p(int i) {
    System.out.println(i);
  }
}
```

Output : 10
20.0

Method Overriding

- Method overriding is when one of the methods in the super class is redefined in the sub-class.

  Provide a new implementation for a method in the subclass.

- The overridden method can widen the accessibility but not narrow it.

  If it is private in the parent class, the child class can make it public but not vice versa.

- The method signature remains the same.

  Method name, parameter list and return type have to match exactly.

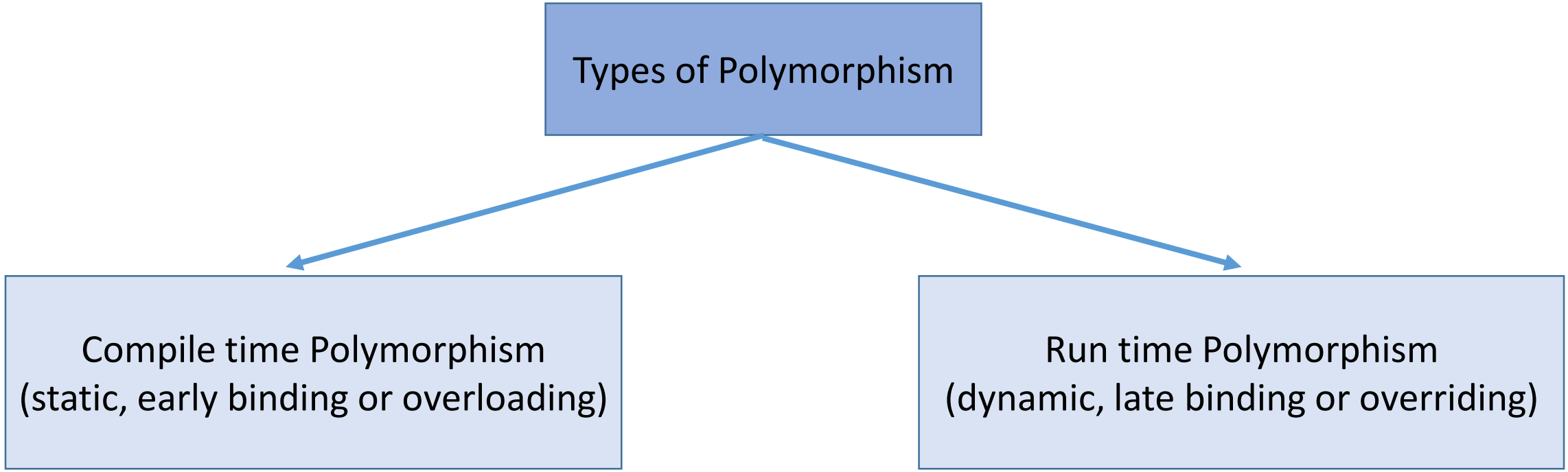
```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
    a.p(10.0);
  }
}

class B {
  public void p(double i) {
    System.out.println(i * 2);
  }
}

class A extends B {
  // This method overrides the method in B
  public void p(double i) {
    System.out.println(i);
  }
}
```

Output: 10.0
10.0

Types of Polymorphism



Compile time Polymorphism

- Also known as static, early binding or overloading.
- Here we have two definitions of the same method p().
- Which p method would be called is determined by the parameter list at the compile time.
- This is the reason for knowing this as compile time polymorphism.

```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
    a.p(10.0);
  }
}

class B {
  public void p(double i) {
    System.out.println(i * 2);
  }
}

class A extends B {
  // This method overloads the method in B
  public void p(int i) {
    System.out.println(i);
  }
}
```

Run time Polymorphism

- Also known as dynamic, late binding or overriding.
- The call to an overridden method is resolved at runtime.
- The child class is overriding the method myMethod() of parent class.
- The child class object assigned to the parent class reference.
- To determine which method would be called, the type of the object would be determined at run-time by JVM.

  Since the object belongs to the child class, the child class version of myMethod() is called.

```
class ABC{
  public void myMethod(){
    System.out.println("Overridden Method");
  }
}

public class XYZ extends ABC{

  public void myMethod(){
    System.out.println("Overriding Method");
  }

 public static void main(String args[]){
            ABC obj = new XYZ();
            obj.myMethod();
  }
}
```

Exercise

- Compare and contrast the runtime polymorphism and compile-time polymorphism.

END