Object Oriented Programming using Java 4

Inheritance & Dynamic Method Dispatch

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

Inheritance
- Inheritance and Private Members
- Referencing Sub-Class Objects

Super keyword

Constructor Call-Order

Method Overriding
- Super and Method Overriding
- Overriding versus Overloading

Dynamic Method Dispatch

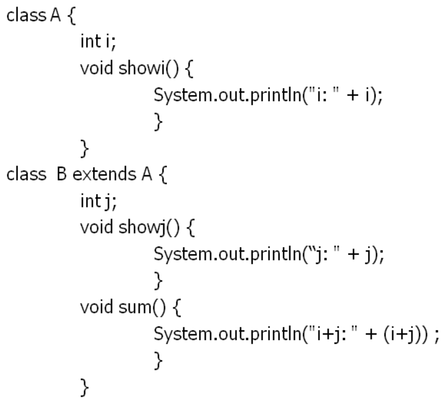
Run-Time Polymorphism

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Inheritance

Inheritance

- One of the pillars of object-orientation
  class sub-class extends super-class { ... }
- Each class has at most one super-class; no multi-inheritance in Java. *No class is a sub-class of itself*



```
class A {
          int i;
          void showi() {
                    System.out.println("i: " + i);
                    }
          }
class  B extends A {
          int j;
          void showj() {
                    System.out.println(“j: " + j);
                    }
          void sum() {
                    System.out.println("i+j: " + (i+j)) ;
                    }
          }
```

Inheritance...

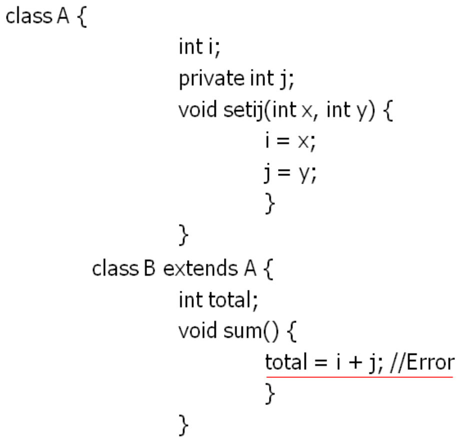
```
class SimpleInheritance {
        public static void main(String args[]) {
                A a = new A();
                B b = new B();
                a.i = 10;
                System.out.println("Contents of a: ");
                a.showi();
                b.i = 7; b.j = 8;
                System.out.println("Contents of b: ");
                b.showi(); b.showj();
                System.out.println("Sum of I and j in b:");
                b.sum();
                }
        }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Inheritance and Private Members

Inheritance and Private Members

- A class may declare some of its members private
- A sub-class has no access to the private members of its super-class



```
class A {
            int i;
            private int j;
            void setij(int x, int y) {
                        i = x;
                        j = y;
                        }
            }
      class B extends A {
            int total;
            void sum() {
                        total = i + j; //Error
                        }
            }
```

Inheritance

Inheritance and Private Members

Referencing Sub-Class Objects

Super keyword

Constructor Call-Order

Method Overriding

Super and Method Overriding

Overriding versus Overloading

Dynamic Method Dispatch

Run-Time Polymorphism

Referencing Sub-Class Objects

Referencing Sub-Class Objects

- A variable of a super-class type may refer to any of its sub-class objects

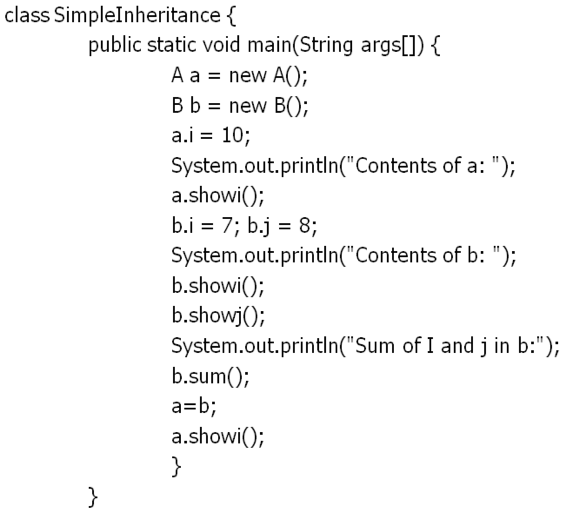
```
class SuperClass{ … }
      class SubClass extends SuperClass{ … }

      SuperClass o1;
      SubClass o2 =  new SubClass();

              o1 = o2;
```

However, the inverse is illegal:

~~o2 = o1;~~

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Referencing Sub-Class Objects...



```
class SimpleInheritance {
        public static void main(String args[]) {
                A a = new A();
                B b = new B();
                a.i = 10;
                System.out.println("Contents of a: ");
                a.showi();
                b.i = 7; b.j = 8;
                System.out.println("Contents of b: ");
                b.showi();
                b.showj();
                System.out.println("Sum of I and j in b:");
                b.sum();
                a=b;
                a.showi();
                }
        }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Super keyword

Super keyword

- Whenever a subclass needs to refer to its immediate super class, it can do so by use of the super keyword
- Super as a Constructor
    - Calling a constructor of a super-class from the constructor of a sub-class:

      super(parameter-list);

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Super keyword...

```
class A {
        int i;
        A(int a)  {    i=a;  }
        void showi() {
                System.out.println("i: " + i);
                }
        }
class  B extends A {
        int j;
        B(int a, int b) {
                super(a);
                j=b;
                }
        void showj() {
                System.out.println("j: " + j);
                }
        void sum() {
                System.out.println("i+j: " + (i+j)) ;
                }
        }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Super keyword...

```
class SimpleInheritance {
        public static void main(String args[]) {
                A a = new A(10);
                B b = new B(7, 8);
                System.out.println("Contents of a: ");
                a.showi();
                System.out.println("Contents of b: ");
                b.showi();
                b.showj();
                System.out.println("Sum of i and j in b:");
                b.sum();
                }
        }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Super keyword...

Super keyword...

- Referencing Sub-Class Objects

```
class  B extends A {
          int j;
          B(int a, int b) {
                    super(a);
                    j=b;
                    }
          B(B ob){
                    super(ob.i);
                    j=ob.j;
                    }
          void showj() {
                    System.out.println(“j: " + j);
                    }
          void sum() {
                    System.out.println("i+j: " + (i+j));
                    }
          }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Super keyword...

```
class SimpleInheritance {
        public static void main(String args[]) {
                A a = new A(10);
                B b = new B(7, 8);
                B c = new B(b);
                System.out.println("Contents of a: ");
                a.showi();
                System.out.println("Contents of b: ");
                b.showi(); b.showj();
                System.out.println("Sum of i and j in b:");
                b.sum();
                c.showi(); c.showj();
                }
        }
```

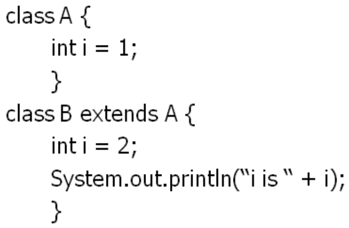
Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Super keyword...

Super keyword...

- Accessing Super-class members

  super.member



```
class A {
      int i = 1;
      }
class B extends A {
      int i = 2;
      System.out.println("i is " + i);
      }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Super keyword...

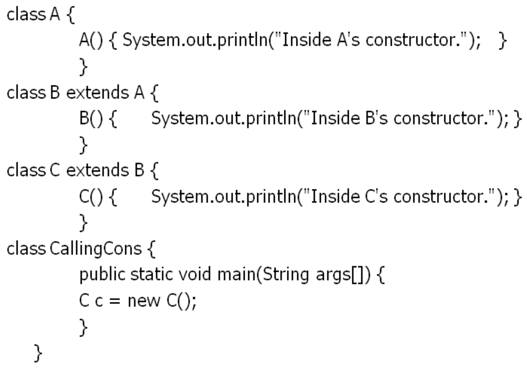
```
class A {
                    int i;
                    }
          class B extends A {
                    int i;
                    B(int a, int b) {     super.i = a; i = b;   }
                    void show() {
                              System.out.println("i in superclass: " + super.i);
                              System.out.println("i in subclass: " + i);
                              }
                    }
class UseSuper {
          public static void main(String args[]) {
                    B subOb = new B(1, 2);
                    subOb.show();
                    }
          }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Constructor Call-Order

Constructor Call-Order

- first call super-class constructors
- then call sub-class constructors
- *In the sub-class constructor, if super(...) is not used explicitly, Java calls the default, parameter-less super-class constructor*



```
class A {
        A() { System.out.println("Inside A's constructor.");   }
        }
class B extends A {
        B() {     System.out.println("Inside B's constructor."); }
        }
class C extends B {
        C() {     System.out.println("Inside C's constructor."); }
        }
class CallingCons {
        public static void main(String args[]) {
        C c = new C();
        }
   }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Method Overriding

Method Overriding

- When a method of a sub-class has the same name and type as a method of the super-class, we say that this method is overridden
- When an overridden method is called from within the sub-class:
  - it will always refer to the sub-class method
  - super-class method is hidden

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Method Overriding...

```
        class A {
            int i, j;
            A(int a, int b) {    i = a; j = b;        }
            void show() { System.out.println("i and j: " + i + " " + j);  }
          }
    class B extends A {
            int k;
            B(int a, int b, int c) {
                    super(a, b);
                    k = c;
                    }
            void show() {      System.out.println("k: " + k);         }
            }
    class Override {
            public static void main(String args[]) {
                    B subOb = new B(1, 2, 3);
                    subOb.show();
                    }
            }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Super and Method Overriding

Super and Method Overriding

- The hidden super-class method may be invoked using super

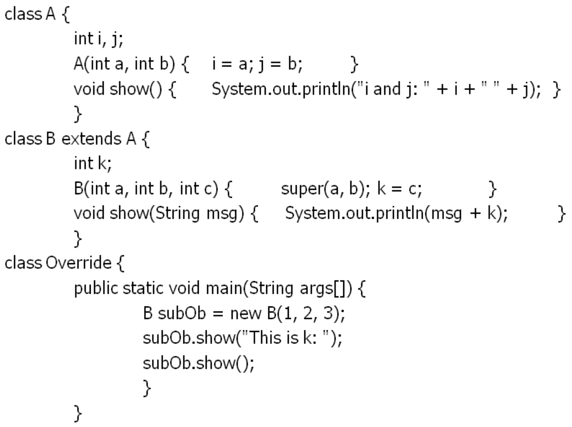
```
class B extends A {
        int k;
        B(int a, int b, int c) {
                super(a, b);
                k = c;
                }
        void show() {
                super.show();
                System.out.println("k: " + k);
                }
        }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Overriding versus Overloading

Overriding versus Overloading

- It occurs only when the names and types of the two methods (super- and sub-class methods) are identical
- If not identical, the two methods are simply overloaded



```
class A {
          int i, j;
          A(int a, int b) {    i = a; j = b;        }
          void show() {      System.out.println("i and j: " + i + " " + j);  }
          }
class B extends A {
          int k;
          B(int a, int b, int c) {        super(a, b); k = c;           }
          void show(String msg) {     System.out.println(msg + k);        }
          }
class Override {
          public static void main(String args[]) {
                    B subOb = new B(1, 2, 3);
                    subOb.show("This is k: ");
                    subOb.show();
                    }
          }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Dynamic Method Dispatch

Dynamic Method Dispatch

- Overriding is the basis for dynamic method dispatch - a call to an overridden method is resolved at run-time, rather than compile-time
- Method overriding allows for dynamic method invocation:
  - an overridden method is called through the super-class variable
  - Java determines which version of that method to execute based on the type of the referred object at the time the call occurs
  - when different types of objects are referred, different versions of the overridden method will be called

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Dynamic Method Dispatch...

```
class A {
        void callme() {
                System.out.println("Inside A's callme method");
                }
        }
class B extends A {
        void callme() {
                System.out.println("Inside B's callme method");
                }
        }
class C extends A {
        void callme() {
                System.out.println("Inside C's callme method");
                }
        }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Dynamic Method Invocation...

```
class Dispatch {
          public static void main(String args[]) {
          A a = new A();
          B b = new B();
          C c = new C();

          A r;
          r = a; r.callme();
          r = b; r.callme();
          r = c; r.callme();
          }
    }
```

Run-Time Polymorphism

```
class Figure {
        double dim1;
        double dim2;
        Figure(double a, double b) {
                dim1 = a; dim2 = b;
                        }
        double area() {
                System.out.println("Area is undefined.");
                return 0;
                }
        }
class Rectangle extends Figure {
        Rectangle(double a, double b) {
                super(a, b);
                }
        double area() {
                System.out.println("Inside Area for Rectangle.");
                return dim1 * dim2;
                }
        }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism

Run-Time Polymorphism...

```
class Triangle extends Figure {
        Triangle(double a, double b) {
                super(a, b);
                }
        double area() {
                System.out.println("Inside Area for Triangle.");
                return dim1 * dim2 / 2;
                }
        }
class FindAreas {
        public static void main(String args[]) {
                Figure f = new Figure(10, 10);
                Rectangle r = new Rectangle(9, 5);
                Triangle t = new Triangle(10, 8);
                Figure figref;
                figref = r; System.out.println(figref.area() );
                figref = t; System.out.println(figref.area() );
                figref = f; System.out.println(figref.area() );
                }
        }
```

Inheritance
Inheritance and Private Members
Referencing Sub-Class Objects
Super keyword
Constructor Call-Order
Method Overriding
Super and Method Overriding
Overriding versus Overloading
Dynamic Method Dispatch
Run-Time Polymorphism