Shreyas Sawant
D7A-55

AIM: To demonstrate the usage of keyword super and final

THEORY

- Keyword super

Whenever a subclass needs to refer to its immediate superclass, it can do so by use of keyword 'super'.
super has two general forms. The first calls the superclass's constructor. The second is used to access a member of the superclass that has been hidden by a member of a subclass.

1) Using super to call Superclass Constructors

A subclass can call a constructor defined by its superclasses by use of the following form of super:

super (args-list);

Here args-list specifies any arguement needed by the constructor in superclass. super() must always be the first statement executed inside a subclass's constructor.

For eg.

```
class Test extends Exam {
    Test (int grades, int percent) {
      super (grades, percent);
    }
}

class Exam {
    int percent, grade;
    Exam (int a, int b)
    { percent = b; grade = a; }
}
```

Here Test calls super() with arguements grades and percent. which intializes the Exam() constructor values.

Now, since constructors can be overloaded, super() can be ~~used~~ called using any form defined by the superclass.

2) Using super to access member of superclass

The 'super' acts like 'this' keyword, except that it always refers to superclass of subclass in which it is used.

super.member

where, member can either be a method or instance variable.

This form of super is most applicable to situations in which member names of a subclass hide members by the same name in superclass.

For eg

```
class A {
    int i;
}

class B extends A {
    int i;
    B (int a, int b) { ~~i=b,~~ super.i = a; i=b; }
    void show {
        System.out.println(" i in subclass : " + i);
        System.out.println (" i in superclass : " + super.i);
    }
}

class Test {
  public static void main(String args[])
  { B ob= new B(2,3);
      ob.show();
  }}
```

Output:

i in subclass : 3

i in superclass : 2

- Keyword final

The final keyword has three uses: first, it can be used to create the equivalent of a named constant, second is to prevent overriding and third is to prevent inheritance

1) To create an equivalent of a named constant

When a variable is declared with final keyword, its value can't be modified, thus a constant. This also means that you must initialize a final variable.

For eg

```
class Test {
  final int CAP = 3;
  public static void main(String args[])
  {   CAP = 5; }
}
```

The above code will ~~to~~ throw a compile-time error.

2) To prevent overiding

To disallow a method from being overridden, specify final as a modifier at the start of its declaration. Methods declared as final cannot be overriden

For eg

```
class A {
    final void meth() {
        System.out.println ("This is last ");
    }
}

class B extends A {
    void meth() {
        System.out.println ("No"),
    } }
```

The above code will throw an compile time error.

3) To prevent inheritance

By declaring a class as final it prevents it from being inherited. However since an abstract class is incomplete by itself and relies upon its subclasses to provide complete implementations, final and abstract cannot be used together to declare a class.

For eg

```
final class A {
  int a; void show () { System.out.println (" Great show"); }
}

class B extends A {
  void err ()
  { System.out.println ("Error "); }
}
```

The above code will show a compile time error.

CONCLUSION:

Errors encountered:

1) Incorrect call to superclass constructor:

```
Child (int a, int b)
{ i=1; j=2;
  super (a,b);
}
```

Solution Call to super must be the first statement in constructor:

```
Child (int a, int b)
{ super (a,b);
  ....
}
```

2) Incorrect call to instance variable of superclass

```
i= i+ super (i);
```

Solution Correct syntax: i= i + super. i;

LAB 8: DEMONSTRATE USAGE OF KEYWORD SUPER AND FINAL

Name: Shreyas Sawant Div: D7A Roll No.: 55

Q.1 Demonstrate any two uses of keyword super.

CODE:

```
class Parent
{       public int i,j;
        Parent(int a,int b)
        {
                i=a;
                j=b;
        }
        void show()
        {
                System.out.println("Show from superclass (i and j): "+i+" "+j+"\n");
        }
}

class Child extends Parent
{       int i,j;
        Child(int a,int b)
        {       super(a,b);     //First use of super: To call the constructor of immediate superclass
                i=1;j=2;

        }
        void show()
        {       //Second use of super: To access a member of superclass that has been hidden by a member of subclass

                i=i+super.i;
                j=j-super.j;
                System.out.println("Show from subclass (i and j): "+i+" "+j);
                super.show();
        }
```

```
}

class Child extends Parent
{       int i,j;
        Child(int a,int b)
        {       super(a,b);     //First use of super: To call the constructor of immediate superclass
                i=1;j=2;

        }
        void show()
        {       //Second use of super: To access a member of superclass that has been hidden by a member of subclass

                i=i+super.i;
                j=j-super.j;
                System.out.println("Show from subclass (i and j): "+i+" "+j);
                super.show();
        }
}
class superUse
{       public static void main(String args[])
        {
                Child ob=new Child(5,7);
                Child ob1=new Child(75,5);
                ob.show();
                ob1.show();
        }
}
```

OUTPUT:

```
C:\Windows\System32\cmd.exe

C:\Users\user\Desktop\SHREYAS\Java Programs\LAB 08>javac superUse.java

C:\Users\user\Desktop\SHREYAS\Java Programs\LAB 08>java superUse
Show from subclass (i and j): 6 -5
Show from superclass (i and j): 5 7

Show from subclass (i and j): 76 -3
Show from superclass (i and j): 75 5


C:\Users\user\Desktop\SHREYAS\Java Programs\LAB 08>
```

Q.2 Demonstrate the importance of keyword final in inheritance and method overriding.

CODE:

```
class One
{       //Methods declared as final cannot be overriden

        final public void skill()
        {       System.out.println("Skills are required to access this." );
        }
}

//classes declared as final cannot be inherited
final class Two extends One
{       //Show error as method cannot be overriden

        void skill()
        {
                System.out.println("Cannot access");
        }

}

class Three extends Two
{
        void skill()
        {       System.out.println("Cannot access it.");
                super.skill();
        }
```

```
        void skill()
        {
                System.out.println("Cannot access");
        }

}

class Three extends Two
{
        void skill()
        {       System.out.println("Cannot access it.");
                super.skill();
        }

}

//final
class finalUse
{
        public static void main(String args[])
        {       Three ob=new Three();
                ob.skill();

        }
}
```

OUTPUT

```
C:\Windows\System32\cmd.exe

C:\Users\user\Desktop\SHREYAS\Java Programs\LAB 08>javac finalUse.java
finalUse.java:21: error: cannot inherit from final Two
class Three extends Two
                    ^
finalUse.java:14: error: skill() in Two cannot override skill() in One
        void skill()
             ^
  overridden method is final
finalUse.java:23: error: skill() in Three cannot override skill() in One
        void skill()
             ^
  overridden method is final
3 errors

C:\Users\user\Desktop\SHREYAS\Java Programs\LAB 08>
```