Department of
Computer Science and Engineering

Title: Inheritance

Object Oriented Programming Lab
CSE 202

Green University of Bangladesh

1 Objective(s)

- Understanding to use Inheritance using Java
- Role of constructors
- Implementing certain types of inheritances
- Access modifiers

2 Problem analysis

In object oriented programming, inheritance is the mechanism by which we attain reusability of certain components through the means of using classes. The process fundamentally being building a structure by making a class and using this class for making other classes while retaining certain portions of the super class(i.e the one we modeled our new class upon). This concept varies from one programming language to another. Here, we will only focus on java progamming language based implementation of the concept of inheritance. For this at first we need to understand some terms such as:

- super class
- constructors
- subclass
- multilevel inheritance
- multiple inheritance

3 Terms

3.1 Superclass

Simply put super class is a kind of class that other class inherits from. Generally most commonly used methods and attributes are part of it. There ara also special kinds of superclasses where we only provide guidance for subclasses (classes that inherits this superclass).

```java

/*java class declaration:*/

class MyClass {
    // field, constructor, and
    // method declarations
}
```

3.2 Subclass

A class that inherits properties from another class are called as subclass. It is often the case that a subclass inherits another subclass. If class B inherits from class A, then A is the superclass and B is the subclass.

```java
/*java subclass declaration:*/
class MyClass extends MySuperClass {
    // field, constructor, and
    // method declarations
}
```

3.3 Constructor

Constructors initializes an object when it is created. A constructors is a methods that has the same name as the class itself. All the supeclass and subclass can have constructor.

```
/*java declaration:*/
class MyClass {
    //  constructor
    Myclass(parameters){
    //
    }
}
```

3.4 Multilevel Inheritance

Let's consider 3 classes, A, B, C, then if class B inherits A then it is called single level inheritance. If further C inherits B, then we have 2 levels. this is called multilevel inheritance. We can have more than 2 levels. When we have more than 2 level of inheritance, we call it multilevel inheritance in java.

```
/*java declaration:*/
class A {
    //  constructor
    A(parameters){
    //
    }
}
class B extends A{
//1st level
}
class C extends B{
//2nd level
}
```

3.5 Multiple Inheritance

Again assume 3 classes: A, B, C. If class B inherits from both A and C we call it multiple inheritance.

```
/*java declaration:*/
class A {
    //  constructor
    A(parameters){
    //
    }
}
class C{
    C(parameters){
    //
    }
}
class B extends A,C{
//
}
```

3.6 Access Classifiers

Summarization:

Location	Private	No Modifier	Protected	Public
same class	yes	yes	yes	no
same package subclass	no	yes	yes	yes
same package non-subclass	no	yes	yes	yes
different package subclass	no	no	yes	yes
different package non-subclass	no	no	no	yes

Table 1: Access Modifiers in Java

4 Implementation

- Create a java project name Lab(this also creates Lab.java file)
- Create a class under the package lab named Class1 (i.e create Class1.java file)
- Create another class under the package lab named Class2 (i.e create Class2.java file)

Lab.java file code:

```java
/*Lab.java file*/
package Lab;
public class Lab {
    public static void main(String[] args) {
        Class1 cs1=new Class1("String passing.");
        //cs1.show();
        //cs1.show_string();
        Class2 cs2=new Class2("Class2 string passing.",3);
        cs2.show();
        cs2.show_string();
        cs2.show_count();
        //System.out.println(cs2.name);
    }
}
```

Create Class1.java file:

```java
package lab;// this Class1 should be in the same package

public class Class1 {
    String s="";
    String name="Class1 name";

    Class1(String s){
        this.s=s;
    }
    public void show(){
        System.out.println("Class1");
    }
    public void show_string(){
        System.out.println(this.name);
    }
}
```

Create Class2.java file:

```
1  package lab;
2  public class Class2 extends Class1{
3      int count=0;
4  
5      Class2(String s, int count){
6          super(s);
7          this.count=c;
8      }
9      @Override
10     public void show(){
11         System.out.println("Class2.");
12         System.out.println(super.name);
13         super.show_string();
14     }
15     public void show_count(){
16         System.out.println("Class2 "+this.count);
17     }
18 }
```

5 Input/Output

Output of the program is given below.

```
Class2.
Class1 name
Class1 name
Class1 name
Class2 3
```

6 Discussion

This is a single level inheritance example. Class1 is inherited by Class2. Class2 uses keyword "super" in constructor to initialize superclass constructor. In Class2, we additionally passed extra parameters.

N.B. Very basic demonstration

7 Lab Task (Please implement yourself and show the output to the instructor)

1. Specific problem/problems provided by the instructor.

7.1 Problem analysis

- Change line 7 in Class2.java to "count=count". What would be the complications and role of keyword "this"?
- What does line 6 in Class2.java do (i.e the role of super)?
- How to handle constructors of superclass in subclasses?

- How to define constructors if we have more than 2 levels of inheritance?

8 Lab Exercise (Submit as a report)

Specific problem/problems for lab report may be provided by course teacher. Some suggested problems:

- Implement Multiple inheritance. 3 Classes A,B,C. Class C inherits both A and B.
- Try various combinations of **public, private, protected** and verify if it satisfies the table in table 1.

9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected. Any other policy applied by course teacher may hold applicable.

10 References

https://docs.oracle.com/javase/tutorial/java/javaOO/classdecl.html

11 Appendix

If need be, course instructor might provide some additional materials.