Implementation of the Inheritance Concept in JAVA

Student's Name

Institutional Affiliation

Implementation of the Inheritance Concept in JAVA

Introduction

In programming, numerous cases exist where different types of objects have shared attributes. The objects with related characteristics can be grouped into one class that describes properties common to all the entities (Budd, 2008). For instance, in a given company, accountants, auditors, finance managers, human resource managers, etc., share the characteristics of an employee, although each has a different role in the organization. Inheritance, an essential feature of object-oriented programming, is a concept that allows the definition of a single class that contains the shared attributes of different but related objects.

Inheritance in JAVA

In the JAVA programming language, Inheritance is used to establish relationships between different classes. There are two main types of classes: parent (also known as the superclass) and child class (also known as subclass). The Parent-class contains properties that a child-class can inherit (Litvin & Litvin, 2010). According to the JAVA syntax, a child class can only be directly linked to one superclass. However, a superclass can have unlimited child classes. Basically, the superclass is at the top of the class hierarchy, while the subclass is at the bottom.

When a child class is developed using the blueprint defined in the superclass, either all or part of the parent class methods and variables are inherited. All public variables, and public static and abstract methods are inherited by the child-class (T. Wu, 2008). However, private methods and variables, private static methods, and constructors in the parent class are not inherited by the child-class and cannot be directly accessed (Litvin & Litvin, 2010). The JAVA syntax allows only indirect reference to private methods and variables of the parent-class.

Implementation of Inheritance in JAVA

Implementation of Inheritance in JAVA uses the *extends* keyword. If *X* is the superclass, and *XY* is the subclass, then the syntax for implementing Inheritance in JAVA is illustrated below.

```
class XY extends X
{

}
```

In the above illustration, the child-class, XY, inherits all the characteristics defined in the parent-class, X.

Inheritance Example

This example illustrates the use of Inheritance in the development of an employee management system. Employees in the company have shared attributes. Therefore, a superclass defines the properties common to all the employees that a child class can inherit.

```
//Definition of the superclass, employee, that contains all the common
properties of employees
class Employee {
   String CompanyDesignation = "Employee";
   String Department = "Finance";

}
//Inheritance of the properties defined in the superclass by the child class
FinanceDptEmployee.
public class FinanceDptEmployee extends Employee{
   String MainRole = "Auditor";
   public static void main(String args[]){
      FinanceDptEmployee obj = new FinanceDptEmployee();
      System.out.println("Department: "+obj.Department);
      System.out.println("Designation: "+obj.CompanyDesignation);
      System.out.println("Departmental Role: "+obj.MainRole);

   }
}
```

In the above implementation, the superclass, *Employees,* contains a definition of the properties, *CompanyDesignation* and *Department,* that are common to all the company employees. The child class *FinanceDptEmployee* inherits the attributes defined in the *Employees* superclass. Moreover, the child class inherits all the methods and attributes defined in the superclass since they are declared as public.

Conclusion

Inheritance is heavily used in the JAVA programming language for code reusability. The concept facilitates the definition of features of a group of related objects – a superclass – and allows the development of new classes – a subclass – based upon properties of the superclass. The use of inheritance in JAVA generally increases the readability of a source code.

References

Budd, T. (2008). *Introduction to object-oriented programming*. Pearson Education.

Litvin, M., & Litvin, G. (2010). *Java methods: Object-oriented programming and data structures*. Skylight Publishing.

Wu, T. (2008). *A Comprehensive introduction to object-oriented programming with Java*. McGraw Hill Higher Education.