CSE 271 Lab 6 – Inheritance & Polymorphism
Spring 2022
Assigned: 3/3/2022
Due: 3/6/2022

Introduction:

In this lab, we will explore inheritance in detail and practice its implementation. You will implement multiple classes in two separate problems to create two unique inheritance hierarchies. You will also write JavaDoc comments for the classes and methods. You must use the *instanceof* operator in the equals methods to ensure there will not be any ClassCastExceptions when you cast objects. Create a project in Eclipse named **Lab6** and complete the following two problems.

Problem 1:

In the first inheritance hierarchy, we will be exploring a work environment that consists of multiple classes. These classes include a Person, Employee, Manager, and Executive. The Employee is a child of Person, Manager is a child of Employee, and Executive is a child of Manager. This problem will help us practice pure inheritance principles. The following are the requirements for each class.

1. **Class Person:** Create a class named **Person** which has the following instance properties and public interface.
    a. Instance Properties:
        i. String name – The name of the person.
        ii. int yearOfBirth – The year of birth of the person.
    b. Public Interface:
        i. **public Person() –** The default constructor which initializes all numbers as 0 and Strings to "".
        ii. **public Person(String name, int yearOfBirth) –** The workhorse constructor.
        iii. Getters and Setters for each instance property.
        iv. **public boolean equals(Object o) –** The method returns true if the two Persons have the same name and year of birth and false, otherwise.
        v. **public String toString() –** The method returns a String representing the Person object that includes the name and year of birth. The formatted String should follow this example: "I am a person whose name is **James Bond** and was born in **1997**.".
2. **Class Employee:** Create a class named **Employee** which is a child of **Person** and has the following instance properties and public interface.
    a. Instance Properties:
        i. double salary – The salary of the person (for example, $85000.00).
    b. Public Interface:
        i. **public Employee() –** The default constructor which initializes all numbers to 0 and Strings to "".
        ii. **public Employee(String name, int yearOfBirth, double salary) –** The workhorse constructor. Note, you must utilize the **super()** constructor call appropriately to set the parent's instance properties.
        iii. Getters and Setters for each instance property.

iv. **public boolean equals(Object o) –** The method returns true if the two Employees have the same name, year of birth, and salary and false, otherwise. Do not forget to use the parent's equals() method when applicable.
v. **public String toString() –** The method returns a String representing the Employee object that includes the name, year of birth, and salary. The formatted String should follow this example: "I am an employee and have a salary of **$50000.00**. I am a person whose name is **James Bond** and was born in **1997**.". Note that you should utilize the parent's toString() method where applicable.

3. **Class Manager:** Create a class named **Manager** which is a child of **Employee** and has the following instance properties and public interface.
    a. Instance Properties:
        i. String department – The department of the manager.
    b. Public Interface:
        i. **public Manager() –** The default constructor which initializes all numbers to 0 and Strings to "".
        ii. **public Manager(String name, int yearOfBirth, double salary, String department) -** The workhorse constructor. Note, you must utilize the **super()** constructor call appropriately to set the parent's instance properties.
        iii. Getters and Setters for each instance property.
        iv. **public boolean equals(Object o) –** The method returns true if the two Managers have the same name, year of birth, salary, and department and false, otherwise. Do not forget to use the parent's equals() method when applicable.
        v. **public String toString() –** The method returns a String representing the Manager object that includes the name, year of birth, salary, and department. The formatted String should follow this example: "I am a manager of the **Human Resource** department. I am an employee and have a salary of **$50000.00**. I am a person whose name is **James Bond** and was born in **1997**.". Note that you should utilize the parent's toString() method where applicable.
4. **Class Executive:** Create a class named **Executive** which is a child of **Manager** and has the following instance properties and public interface.
    a. Instance Properties:
        i. String officeLocation – The office location of the executive (i.e., "suite 300, room 419").
    b. Public Interface:
        i. **public Executive() –** The default constructor which initializes all numbers to 0 and Strings to "".
        ii. **public Executive(String name, int yearOfBirth, double salary, String department, String officeLocation) -** The workhorse constructor. Note, you must utilize the **super()** constructor call appropriately to set the parent's instance properties.
        iii. Getters and Setters for each instance property.
        iv. **public boolean equals(Object o) –** The method returns true if the two Executives have the same name, year of birth, salary, department, and officeLocation and false, otherwise. Do not forget to use the parent's equals() method when applicable.
        v. **public String toString() –** The method returns a String representing the Executive object that includes the name, year of birth, salary, department, and officeLocation. The formatted String should follow this example: "I am an executive and my office location is **room 419**. I am a manager of the **Human Resource** department. I am an employee and have a salary of **$50000.00**. I am a

person whose name is **James Bond** and was born in **1997**.". Note that you should utilize the parent's toString() method where applicable.

Problem 2:

In the second inheritance hierarchy, we will be creating animals that consists of a smaller hierarchy than the previous problem. These classes include an Animal, Dog, and Cat. The Dog and Cat classes are both children of the Animal class. This problem will help us practice our inheritance and polymorphism principles. The following are the requirements for each class.

1. **Class Animal:** Create a class named **Animal** that has the following instance properties and public interface.
    a. Instance Properties:
        i. int age – the age of the animal.
        ii. String size – the size of the animal.
    b. Public Interface:
        i. **public Animal() –** The default constructor which initializes all numbers as 0 and Strings to "".
        ii. **public Animal(int age, String size) –** The workhorse constructor.
        iii. Getters and Setters for each instance property.
        iv. **public boolean equals(Object o) –** The method returns true if the two Animals have the same age and size and false, otherwise.
        v. **public String toString() –** The method returns a String representing the Animal object that includes the age and size. The formatted String should follow this example: "I am an unknown **medium** animal that is **9** years old.".
        vi. **public String speak()** – This method returns a String representing an unknown Animal. Specifically, it returns the String "Unintelligible sound".
2. **Class Dog:** Create a class named **Dog** that is a child of **Animal** and has the following instance properties and public interface.
    a. Instance Properties:
        i. String breed – the breed of the dog.
    b. Public Interface:
        i. **public Dog() –** The default constructor which initializes all numbers to 0 and Strings to "".
        ii. **public Dog(int age, String size, String breed) -** The workhorse constructor. Note, you must utilize the **super()** constructor call appropriately to set the parent's instance properties.
        iii. Getters and Setters for each instance property.
        iv. **public boolean equals(Object o) –** The method returns true if the two Dogs have the same age, size, and breed and false, otherwise. Do not forget to use the parent's equals() method when applicable.
        v. **public String toString() –** The method returns a String representing the Dog object that includes the age, size, and breed. The formatted String should follow this example: "I am a/an **large husky** breed dog that is **5** years old.".
        vi. **public String speak()** – This method returns a String representing a Dog. Specifically, it returns the String "Bark".
3. **Class Cat:** Create a class named **Cat** that is a child of **Animal** and has the following instance properties and public interface.
    a. Instance Properties:
        i. int numberOfLives – the number of lives the Cat has left.

b. Public Interface:
    i. **public Cat() –** The default constructor which initializes all numbers to 0 and Strings to "".
    ii. **public Cat(int age, String size, int numberOfLives) -** The workhorse constructor. Note, you must utilize the **super()** constructor call appropriately to set the parent's instance properties.
    iii. Getters and Setters for each instance property.
    iv. **public boolean equals(Object o) –** The method returns true if the two Cats have the same age, size, and numberOfLives and false, otherwise. Do not forget to use the parent's equals() method when applicable.
    v. **public String toString() –** The method returns a String representing the Cat object that includes the age, size, and numberOfLives. The formatted String should follow this example: "I am a **small** cat that is **2** years old and have **9** lives left.".
    vi. **public String speak()** – This method returns a String representing a Cat. Specifically, it returns the String "Meow".

Tester Class:

In the main method of your Tester class, you must test the classes from both problems listed above. Because there are no validation checks necessary in this Lab, make sure to simply test your equals(), toString(), constructors, and speak() methods.

Additional Notes:

- Make sure you include JavaDoc comments for all methods and classes including parameter and return descriptions.
- Make sure that all classes are named correctly.
- There is no explicit validation checking needed for this Lab assignment.

Submission Instructions:

After you have completed the lab assignment, locate your source code (**Person.java, Employee.java, Manager.java, Executive.java, Animal.java, Dog.java, and Cat.java**) in your workspace and submit it to the corresponding Lab 6 assignment's CODE plugin.

Rubric:

Task	Grade
Problem 1	
Private instance properties declared	9
Constructors correctly implemented	9
Getters and setters correctly implemented	9
equals() method correctly implemented	9
toString() method correctly implemented	9
Problem 2	
Private instance properties declared	9
Constructors correctly implemented	9
Getters and setters correctly implemented	9
equals() method correctly implemented	9
toString() method correctly implemented	9
speak() method correctly implemented	5
Adequate JavaDoc included for both problems and followed the Miami University coding guidelines	5
Total	**100**