Universidade Católica de Brasília
Disciplina: Programação Orientada a Objetos
Professor(a): Victor Manuel Zerefos de Oliveira
Estudante: Douglas Souza de Lima
Matrícula: UC23200709

Lista 2

1. Crie uma classe base Animal com um método makeSound(). Depois, crie duas subclasses Dog e Cat que herdam de Animal e sobrescrevem o método makeSound() para imprimir sons específicos ("Bark" e "Meow", respectivamente).

Animal.java

```java
package edu.animal;

public abstract class Animal {

    protected abstract void makeSound();

}
```

Cat.java

```java
package edu.animal.feline;

import edu.animal.Animal;

public class Cat extends Animal {

    @Override
    public void makeSound() {
        System.out.println("Meow");
    }

}
```

Dog.java

```java
package edu.animal.canine;

import edu.animal.Animal;

public class Dog extends Animal {

    @Override
    public void makeSound() {
        System.out.println("Bark");
    }

}
```

Main.java

```java
package edu.animal;

import edu.animal.Animal;
import edu.animal.canine.Dog;
import edu.animal.feline.Cat;

public class Main {

    public static void main(String[] args) {
        Animal cat = new Cat();
        System.out.println("A cat sounds like: ");
        cat.makeSound();

        Animal dog = new Dog();
        System.out.println("A dog sounds like: ");
        dog.makeSound();
    }

}
```

2. Crie uma classe base Person com atributos name e age, e um construtor para inicializá-los. Depois, crie uma subclasse Student que herda de Person e adiciona um atributo studentID, e também inicialize esse atributo através do construtor.

Person.java

```java
package edu.person;

public abstract class Person {

    protected String name;
    protected int age;

    protected Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    protected int getAge() {
        return age;
    }

    protected void setAge(int age) {
        this.age = age;
    }

    protected String getName() {
        return name;
    }

    protected void setName(String name) {
        this.name = name;
    }

}
```

Student.java

```
package edu.person.school;

import java.util.Objects;
import edu.person.Person;

public class Student extends Person implements Comparable<Student> {

    private int studentID;

    public Student(String name, int age, int studentID) {
        super(name, age);
        this.studentID = studentID;
    }

    public int getStudentID() {
        return this.studentID;
    }

    public void setStudentID(int studentID) {
        this.studentID = studentID;
    }

    @Override
    public String toString() {
        return "Student [name='" + super.getName() + "', age=" +
super.getAge() + ", studentID=" + this.studentID + "]";
    }

    @Override
    public int hashCode() {
        return Objects.hash(this.studentID);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (!(o instanceof Student)) {
            return false;
        }
        Student s = (Student) o;
        return this.studentID == s.getStudentID();
    }

    @Override
    public int compareTo(Student student) {
        return super.getName().compareToIgnoreCase(student.getName());
    }

}
```

Main.java

```
package edu.person;

import java.util.function.Consumer;
```

```java
import java.util.HashSet;
import edu.person.school.Student;
import java.util.Set;

public class Main {

    public static void main(String[] args) {
        // creates a set of elements of type Student
        // it doesn't allow duplicate elements based on the studentID
attribute
        Set<Student> students = new HashSet<>() {
            {
                add(new Student("Ana", 24, 12));
                add(new Student("Gisele", 19, 10));
                add(new Student("Carlos", 32, 8));
                add(new Student("James", 22, 7));
                add(new Student("Karoline", 23, 3));
                add(new Student("Vanessa", 29, 15));
                add(new Student("Matthew", 27, 9));
                add(new Student("Josh", 26, 5));
                add(new Student("Smith", 28, 13));
                add(new Student("Misha", 30, 2));
            }
        };
        // creates a consumer to print all the elements of type Student
        Consumer<Student> printStudents = student -> {
            System.out.printf("%10d %20s %10d%n", student.getStudentID(),
student.getName(), student.getAge());
        };
        System.out.println();
        System.out.printf("%10s %20s %10s%n", "ID", "NAME", "AGE");
        System.out.printf("%10s %20s %10s%n", "--", "----", "---");
        students.forEach(printStudents);
    }

}
```