Week 5 Homework Q19

Telmen Enkhbold

San Fransico Bay University

CE480 - Java and Internet Application

Dr. Chang, Henry

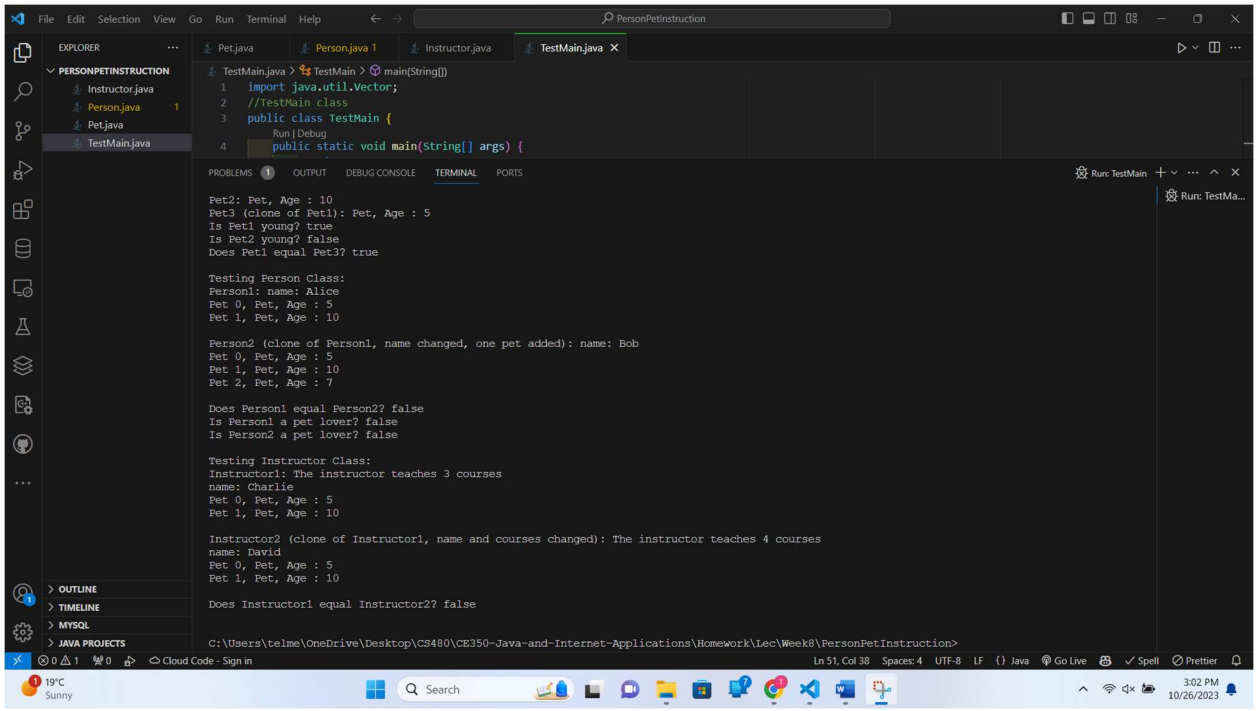
10/12/2023

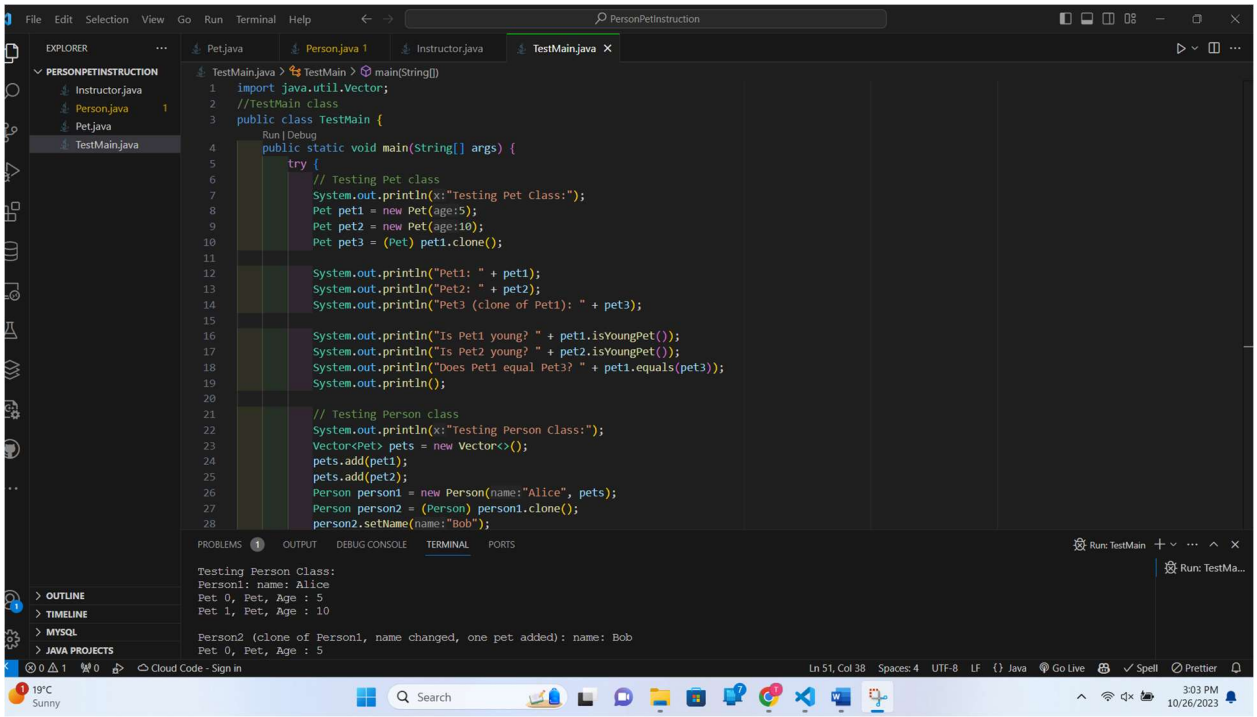
Author Note

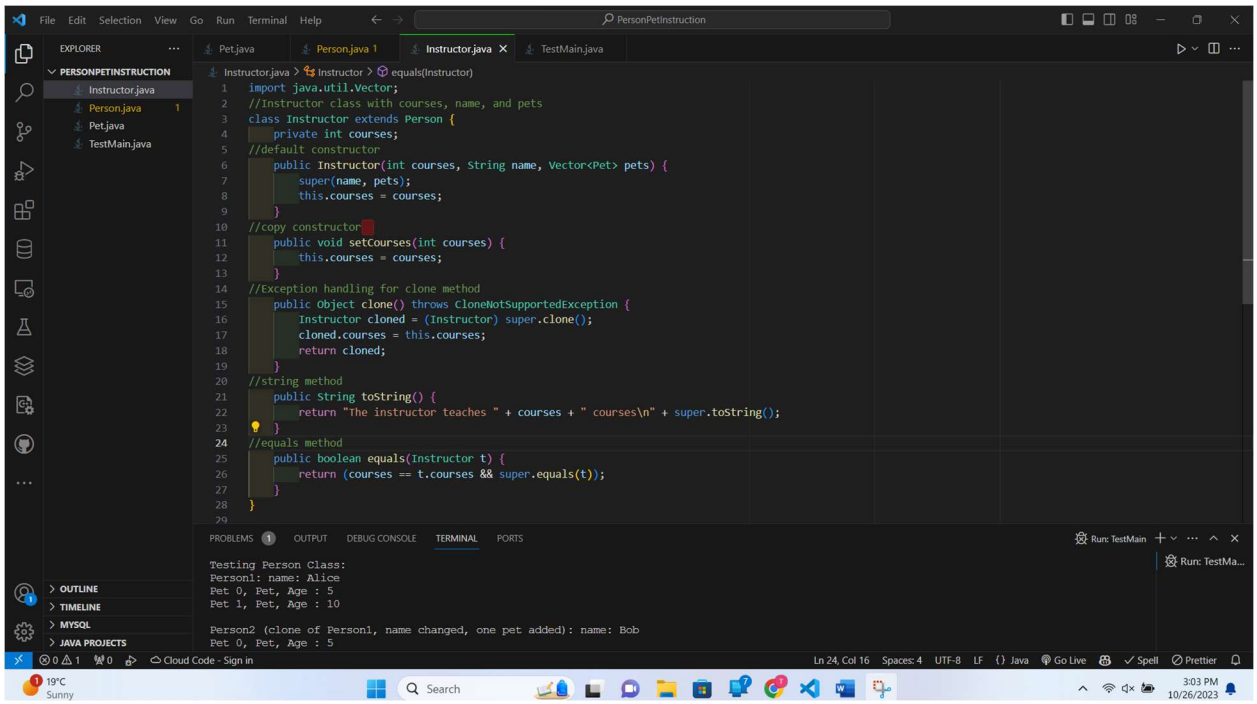
The Question

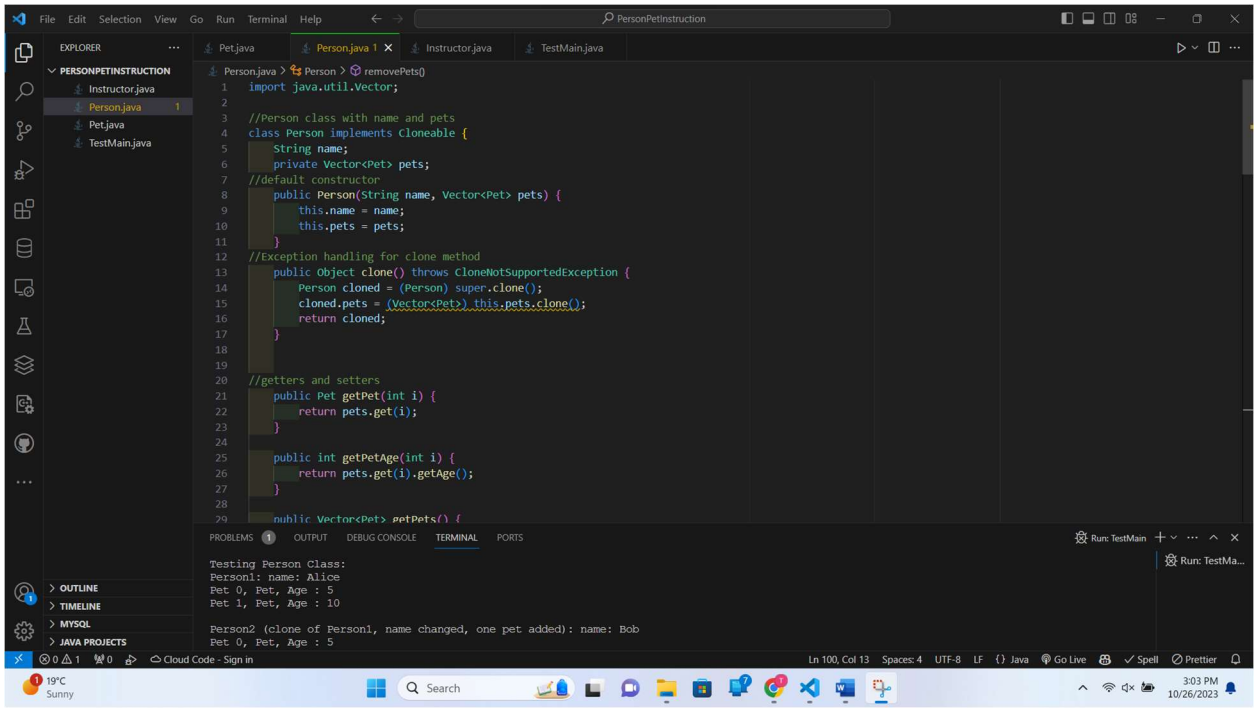
Object-Oriented Programming: Person + Pet + Instructore

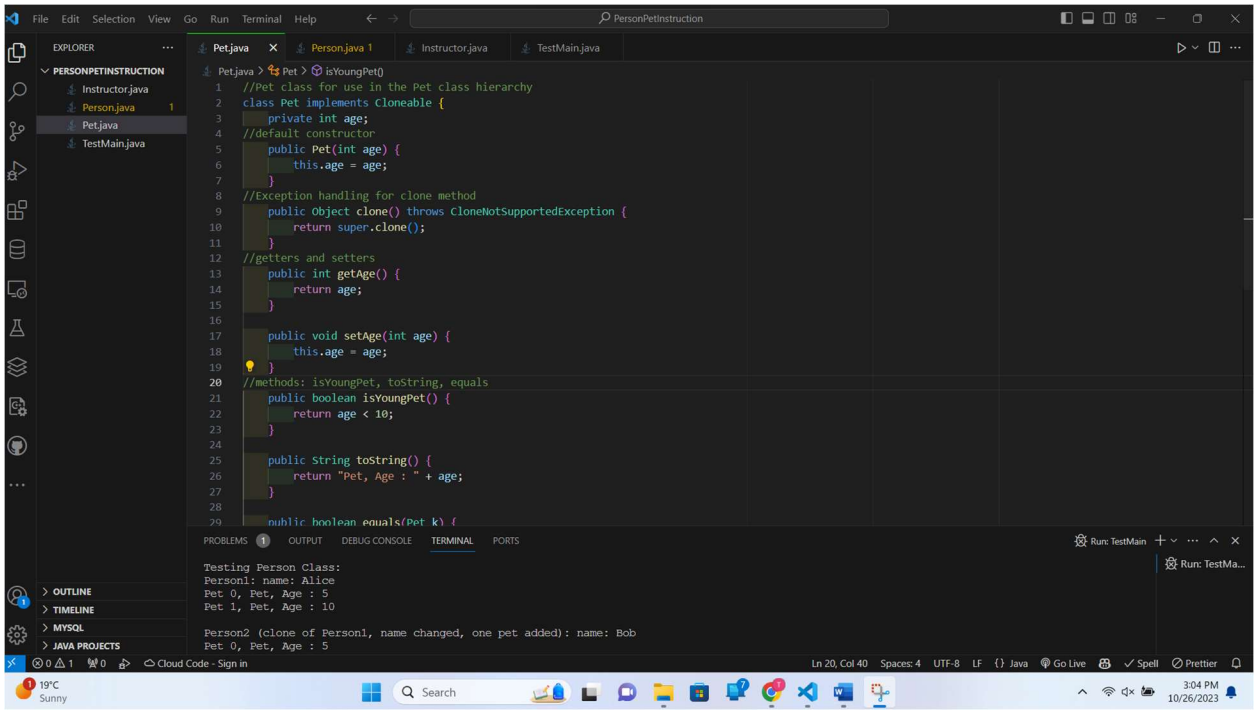
==========ScreenShots==========




TestMain.java
import java.util.Vector;
//TestMain class
public class TestMain {
Run | Debug
public static void main(String[] args) {
try {
// Testing Pet class
System.out.println(x:"Testing Pet Class:");
Pet pet1 = new Pet(age:5);
Pet pet2 = new Pet(age:10);
Pet pet3 = (Pet) pet1.clone();
System.out.println("Pet1: " + pet1);
System.out.println("Pet2: " + pet2);
System.out.println("Pet3 (clone of Pet1): " + pet3);
System.out.println("Is Pet1 young? " + pet1.isYoungPet());
System.out.println("Is Pet2 young? " + pet2.isYoungPet());
System.out.println("Does Pet1 equal Pet3? " + pet1.equals(pet3));
System.out.println();
// Testing Person class
System.out.println(x:"Testing Person Class:");
Vector<Pet> pets = new Vector<>();
pets.add(pet1);
pets.add(pet2);
Person person1 = new Person(name:"Alice", pets);
Person person2 = (Person) person1.clone();
person2.setName(name:"Bob");
Testing Person Class:
Person1: name: Alice
Pet 0, Pet, Age : 5
Pet 1, Pet, Age : 10
Person2 (clone of Person1, name changed, one pet added): name: Bob
Pet 0, Pet, Age : 5


Instructor.java
import java.util.Vector;
//Instructor class with courses, name, and pets
class Instructor extends Person {
private int courses;
//default constructor
public Instructor(int courses, String name, Vector<Pet> pets) {
super(name, pets);
this.courses = courses;
}
//copy constructor
public void setCourses(int courses) {
this.courses = courses;
}
//Exception handling for clone method
public Object clone() throws CloneNotSupportedException {
Instructor cloned = (Instructor) super.clone();
cloned.courses = this.courses;
return cloned;
}
//string method
public String toString() {
return "The instructor teaches " + courses + " courses\n" + super.toString();
}
//equals method
public boolean equals(Instructor t) {
return (courses == t.courses && super.equals(t));
}
}
Testing Person Class:
Person1: name: Alice
Pet 0, Pet, Age : 5
Pet 1, Pet, Age : 10
Person2 (clone of Person1, name changed, one pet added): name: Bob
Pet 0, Pet, Age : 5





==========The Code==========

Instructor.java

```
import java.util.Vector;
//Instructor class with courses, name, and pets
```

```
class Instructor extends Person {
    private int courses;
//default constructor
    public Instructor(int courses, String name, Vector<Pet> pets) {
        super(name, pets);
        this.courses = courses;
    }
//copy constructor
    public void setCourses(int courses) {
        this.courses = courses;
    }
//Exception handling for clone method
    public Object clone() throws CloneNotSupportedException {
        Instructor cloned = (Instructor) super.clone();
        cloned.courses = this.courses;
        return cloned;
    }
//string method
    public String toString() {
        return "The instructor teaches " + courses + " courses\n" +
super.toString();
    }
//equals method
    public boolean equals(Instructor t) {
        return (courses == t.courses && super.equals(t));
    }
}
```

Person.java

```
import java.util.Vector;

//Person class with name and pets
class Person implements Cloneable {
    String name;
    private Vector<Pet> pets;
//default constructor
    public Person(String name, Vector<Pet> pets) {
        this.name = name;
        this.pets = pets;
    }
//Exception handling for clone method
    public Object clone() throws CloneNotSupportedException {
        Person cloned = (Person) super.clone();
        cloned.pets = (Vector<Pet>) this.pets.clone();
```

```
        return cloned;
    }


//getters and setters
    public Pet getPet(int i) {
        return pets.get(i);
    }

    public int getPetAge(int i) {
        return pets.get(i).getAge();
    }

    public Vector<Pet> getPets() {
        return pets;
    }

    public int getNumOfPets() {
        return pets.size();
    }

    public void setPet(int i, int age) {
        pets.get(i).setAge(age);
    }

    public void setPets(Vector<Pet> pets) {
        this.pets = pets;
    }

    public void setName(String name) {
        this.name = name;
    }
//methods: isPetLover, removeAllPets, addPet, insertPet, firstPet,
lastPet, removePet, toString, equals
    public boolean isPetLover() {
        return pets.size() > 3;
    }

    public void removeAllPets() {
        pets.clear();
    }

    public void addPet(int age) {
        pets.add(new Pet(age));
    }
```

```
    public void insertPet(int age, int index) {
        pets.add(index, new Pet(age));
    }

    public Pet firstPet() {
        return pets.firstElement();
    }

    public Pet lastPet() {
        return pets.lastElement();
    }

    public void removePet(int i) {
        pets.remove(i);
    }
//toString and equals methods
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("name: ").append(name).append("\n");
        for (int i = 0; i < pets.size(); i++) {
            sb.append("Pet ").append(i).append(",
").append(pets.get(i).toString()).append("\n");
        }
        return sb.toString();
    }
//equals method
    public boolean equals(Person t) {
        if (!name.equals(t.name)) {
            return false;
        }
        if (getNumOfPets() != t.getNumOfPets()) {
            return false;
        }
        for (int i = 0; i < getNumOfPets(); i++) {
            if (!getPet(i).equals(t.getPet(i))) {
                return false;
            }
        }
        return true;
    }
//Destructor
    public void removePets() {
        pets.clear();
    }
```

```
}
```

Pet.java

```
//Pet class for use in the Pet class hierarchy
class Pet implements Cloneable {
    private int age;
//default constructor
    public Pet(int age) {
        this.age = age;
    }
//Exception handling for clone method
    public Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
//getters and setters
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
//methods: isYoungPet, toString, equals
    public boolean isYoungPet() {
        return age < 10;
    }

    public String toString() {
        return "Pet, Age : " + age;
    }

    public boolean equals(Pet k) {
        return this.age == k.getAge();
    }
}
```

TestMain.java

```
import java.util.Vector;
//TestMain class
public class TestMain {
    public static void main(String[] args) {
        try {
```

```
            // Testing Pet class
            System.out.println("Testing Pet Class:");
            Pet pet1 = new Pet(5);
            Pet pet2 = new Pet(10);
            Pet pet3 = (Pet) pet1.clone();

            System.out.println("Pet1: " + pet1);
            System.out.println("Pet2: " + pet2);
            System.out.println("Pet3 (clone of Pet1): " + pet3);

            System.out.println("Is Pet1 young? " + pet1.isYoungPet());
            System.out.println("Is Pet2 young? " + pet2.isYoungPet());
            System.out.println("Does Pet1 equal Pet3? " +
pet1.equals(pet3));
            System.out.println();

            // Testing Person class
            System.out.println("Testing Person Class:");
            Vector<Pet> pets = new Vector<>();
            pets.add(pet1);
            pets.add(pet2);
            Person person1 = new Person("Alice", pets);
            Person person2 = (Person) person1.clone();
            person2.setName("Bob");
            person2.addPet(7);

            System.out.println("Person1: " + person1);
            System.out.println("Person2 (clone of Person1, name changed,
one pet added): " + person2);
            System.out.println("Does Person1 equal Person2? " +
person1.equals(person2));
            System.out.println("Is Person1 a pet lover? " +
person1.isPetLover());
            System.out.println("Is Person2 a pet lover? " +
person2.isPetLover());
            System.out.println();

            // Testing Instructor class
            System.out.println("Testing Instructor Class:");
            Instructor instructor1 = new Instructor(3, "Charlie", pets);
            Instructor instructor2 = (Instructor) instructor1.clone();
            instructor2.setCourses(4);
            instructor2.setName("David");

            System.out.println("Instructor1: " + instructor1);
```

```
            System.out.println("Instructor2 (clone of Instructor1, name
and courses changed): " + instructor2);
            System.out.println("Does Instructor1 equal Instructor2? " +
instructor1.equals(instructor2));
            System.out.println();

        }
//Exception handling for clone method
        catch (CloneNotSupportedException e) {
            System.out.println("Clone not supported: " + e.getMessage());
        }
    }
}
```

Reference

https://hc.labnet.sfbu.edu/~henry/sfbu/course/introjava/inheritance/slide/exercises4.htmlLinks to an external site.

Q19 ==> Object-Oriented Programming: Person + Pet + Instructor

Github

https://github.com/Georgycas/CE350-Java-and-Internet-

Applications/tree/main/Homework/Lec/Week8/PersonPetInstruction