LP2A Written Exam

Friday April 28th, 2023

Spring 23 Duration 1h30

No documents

Exercise #1 What is appearing on the screen?

Question #1: Analyse the following code and draw the associated class diagram.

Question #2:What this program is printing on the screen?

```
import java.util.LinkedList;

abstract class A {
    public abstract void m();

    public void n() {
        System.out.println("This is a generic object.");
    }
}

class B extends A {
    public void m() {
        System.out.println("Performing action B.");
    }

    public void n() {
        System.out.println("This is an object of type B.");
    }
}

class C extends A {
    public void m() {
        System.out.println("Performing action C.");
    }

    public void n() {
        System.out.println("This is an object of type C.");
    }
}

class D extends C {
    public void n() {
        System.out.println("This is an object of type D.");
    }
}

class E extends A {
    public void m() {
        System.out.println("Performing action E.");
    }

    public void n() {
        System.out.println("This is an object of type E.");
    }
}

class F extends E {
    public void n() {
        System.out.println("This is an object of type F.");
    }
}

public class MainClass {
    public static void main(String[] args) {
        LinkedList<A> objects = new LinkedList<A>();

        objects.add(new B());
        objects.add(new D());
        objects.add(new C() {
            public void m() {
                System.out.println("Performing a custom action.");
            }
        });
        objects.add(new E());
        objects.add(new F());

        for (A object : objects) {
            object.m();
            object.n();
            System.out.println();
        }
    }
}
```

Exercise #2 It reminds me something

In a University you can find different categories of members:

- Student: which can follow a course
- Teacher: which can give a course
- Teacher-researcher: which can give a course and do research
- Researcher: which only do research
- PhD student: which do research and can follow and give a course
- Member of administrative staff: which does not give or follow course and do not do research

  A researcher working in a particular field, a teacher teaches one or more specific matters and a student is enrolled in one or more UV.

Question #1: Propose an organization as classes for these concepts. Use interfaces for the concepts of teaching, doing research and study. What is the solution here abstract classes or interfaces? Explain.

Question #2: First we want to focus on the UV class. Write it while considering that the other class are already available. What kind of data structure we will use for storing the students who are registered to this UV?

Question #3: Write the other classes, abstract classes and/or interfaces you designed in question #1. Pay a particular attention to the toString method each time it is necessary. For your record, you can find below the test class and its on screen result.

Question #4: We now decide that the Administrative staff can follow courses and do lectures. What are the modifications to make in the class architecture and in the code to fit this new rule?

Code

```
import java.util.ArrayList;

public class Test {

     public static void main(String[] args) {

ArrayList<Person> list = new ArrayList<Person>();

Teacher t= new Teacher("John","Smith");

TeacherResearcher t2=new TeacherResearcher("Bob", "Smith",
Topics.AI);

PhDStudent phds= new PhDStudent("Edgard",  "Jean-Henry",
Topics.VirtualReality);

AdministrativeStaff as= new AdministrativeStaff("Sponge",
"Bob Brother");

Researcher r= new
Researcher("Jean","leBon",Topics.Optimization);

list.add(t);
list.add(t2);
list.add(phds);
list.add(as2);
list.add(as);
list.add(r);

UV LP2A = new UV(t,"Object Oriented Programing");
UV LP25 = new UV(t2,"Systems and Shell scripts");

t.teach(LP2A);
t2.teach(LP25);
phds.teach(LP25);

for (int i = 0 ; i <5 ; i++)
         { Student st= new Student("FirstName"+i,"Name"+i*2);
           LP2A.addStudent(st);
           list.add(st);
         }
for (int i = 5 ; i <10 ; i++)
         {Student st= new Student("FirstName"+i,"Name"+i*2);
          LP25.addStudent(st);
           list.add(st);
         }
LP2A.addStudent(as);

System.out.println("Persons' List");

for (Person p: list)
         { System.out.println(p);}
     }
}
```

Screen

```
Persons' List
Name:  Smith  Firstname:  John  taught:  Object  Oriented
Programing
Name: Smith Firstname: Bob taught: Systems and Shell scripts
Research Topics: AI
Name:  Jean-Henry  Firstname:  Edgard  Research  Topics:
VirtualReality taught UV: Systems and Shell scripts
Name: Bob Brother Firstname: Sponge
Name: Bob Firstname: Sponge
Name: leBon Firstname: Jean Research Topics: Optimization
Name: Name0 Firstname: FirstName0
Name: Name2 Firstname: FirstName1
Name: Name4 Firstname: FirstName2
Name: Name6 Firstname: FirstName3
Name: Name8 Firstname: FirstName4
Name: Name10 Firstname: FirstName5
Name: Name12 Firstname: FirstName6
Name: Name14 Firstname: FirstName7
Name: Name16 Firstname: FirstName8
Name: Name18 Firstname: FirstName9
```