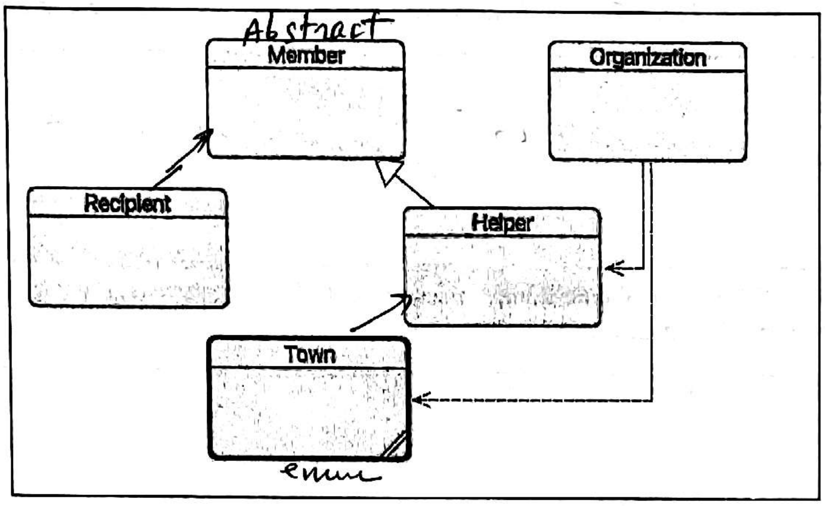
UEF4.3. Object Oriented Programming



Given that the DonationType class implements the donate(String recipientName, String typeContribution) method from the Donation interface, here is an excerpt of the code proposed by the engineer:

class Helper extends Member {
 private String name; //Helper's name
 private int num; // Helper's number
 private final int numCurrent = 0;
 ...// other attributes
 public Helper(String name, boolean isManager) {
 super.name = name;
 num = numCurrent++;
 if (isManager)
 don= new Donation();
 }
public String toString() { return "Helper"; }}
	public abstract class Member {
	 private String name; // Member's name
	 protected Donation don; // Donation Type
	 ...// other attributes
	
	 public Member(String name) {
	 this.name = name;
	 }
	 public void help(String typeContrib) {
	 if (don != null)
	 don.donate(name, typeContrib);
	 }}

2. In the Helper class, four compilation errors are present. Identify these errors and suggest corrections.

Error	Correction

Scanné avec CamScanner

National School of Computer Science (ESI-Algiers) 2CP 2024/2025

UEF4.3. Object Oriented Programming

OOP Test(45mn)

...................Group:...04

Exercise 1: Circle the correct answer(s) (5pts)

Q1. What is the main difference between an interface and an abstract class?

A. An interface can contain private attributes, unlike an abstract class.
[✓ B. An abstract class factors code within a hierarchy, while an interface defines a common contract for classes without requiring a parent-child relationship.]
C. An interface cannot be instantiated, unlike an abstract class.
D. None of the above.

Q2. What happens if two methods in the same class have the same signature but different return types?

A. They are valid if they have compatible return types.
B. Compilation error.
[✓ C. Java selects the method with the most specific return type.]
D. It depends on the compiler.

Q3. What happens if an abstract class in Java contains an abstract method, but a derived class does not override it?

A. Compilation error.
B. The derived class must also be declared abstract.
C. The abstract method is ignored.
D. The program runs normally.

Q4. What happens if a package p1 contains a class A with a protected field, and a class B from another package p2 tries to access it without inheritance?

A. Access is allowed.
[✓ B. Compilation error.]
C. Depends on the CLASSPATH.
D. A warning, but the program compiles.

Q5. If an interface~~,~~ I1 extends an interface I2~~,~~ and ~~a class~~ C implements I1~~, which~~ methods is class C required to implement?

A. Only methods from I1
B. Only those from I2
C. Those from both I1 and I2
[✓ D. None, because an interface cannot inherit from another]

Q6. What will be the output of the following code?

```
class A {
   int x = 10;

   void display() {
      System.out.println(" Class A : x = " + x);
   }
}

class B extends A {
   private int x = 20;

   void display() {
      System.out.println(" Class B : x = " + x);
   }
}

public class Test {
   public static void main(String[] args) {
      A obj = new B();
      System.out.print(obj.x);
      obj.display();
   }
}
```

A. 10 Class A : x = 10
B. 20 Class B : x = 20
[✓ C. 10 Class B : x = 20]
D. 20 Class A : x = 10

UEF4.3. Object Oriented Programming

Q7. What is the difference between overloading and overriding in Java?

A. Overriding consists of redefining a method in a subclass by changing its return type.
[✓ B.] Overriding modifies the behavior of an inherited method while keeping the same signature and return type, whereas overloading defines multiple methods with the same name but different signatures.
C. Overloading and overriding apply only to static methods.
D. Overloading allows extending a method of a class through inheritance, while overriding allows overloading a method only within the same class.

Q8. Why can't an enum in Java inherit from another class?

A. Because an enum is actually a special class that already inherits from java.lang.Enum
B. Because enums are designed to represent constants rather than polymorphic objects
C. Because enums have a special behavior with predefined instances, making inheritance unnecessary
[✓ D.] All of the above

Q9. If we want to use two classes with the same name but from different packages, we can:

A. Import both packages.
[✓ B.] Import the package of one class and use the fully qualified name (package.ClassName) for the other.
[✓ C.] Use the fully qualified name for both classes.
D. All of these options are correct.

Q10. What happens if a class implements two interfaces that have a method with the same signature and return type?

E. Compilation error, even if the interfaces contain only method signatures without implementation.
[✓ F.] Java allows the class to inherit an existing implementation only if the method is defined as default in one of the interfaces.
G. Java arbitrarily chooses one of the interfaces as a reference and ignores the other.
H. Java requires the class to explicitly override the method, even if it is a default method.

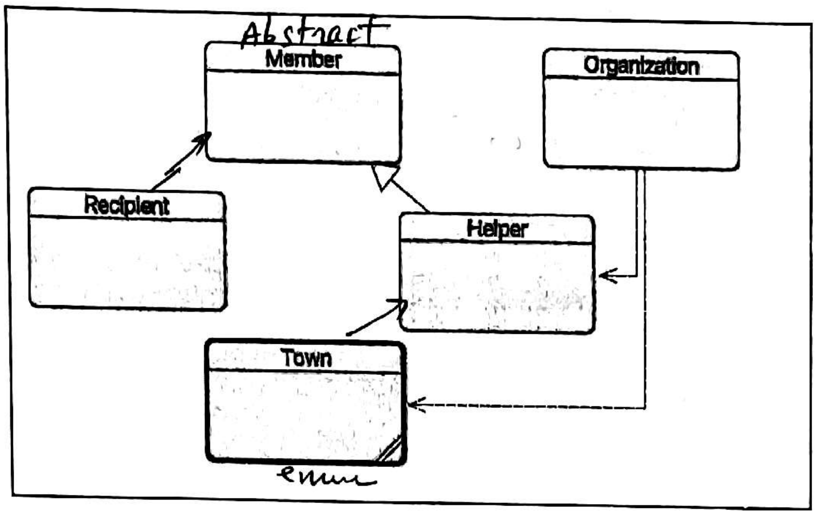
Exercise 2 (5pts)

A humanitarian organization wants to develop a system to manage its members:helpers and recipients. A helper has a first name, last name and town. A recipient has a first name, last name, town and address. The system should allow helpers to donate to recipients and record their contributions.

1. The engineer has identified the classes illustrated in the following diagram. Complete the diagram based solely on the above description:

- Establish the relationships (inheritance, use, and/or implementation) between the different classes.
- Specify their types (abstract classes, interfaces, or enumerations) if necessary.

UEF4.3. Object Oriented Programming



Given that the DonationType class implements the donate(String recipientName, String typeContribution) method from the Donation interface, here is an excerpt of the code proposed by the engineer:

```
class Helper extends Member {
 private String name; //Helper's name
 private int num; // Helper's number
 private final int numCurrent = 0;
 ...// other attributes
   public Helper(String name, boolean isManager) {
         super.name = name;
         num = numCurrent++;
         if (isManager)
             don= new Donation();
         }
public String toString() { return "Helper"; }}
```

```
public abstract class Member {
        private String name; // Member's name
        protected Donation don; // Donation Type
          ...// other attributes

        public Member(String name) {
        this.name = name;
        }
        public void help(String typeContrib) {
         if (don != null)
             don.donate(name, typeContrib);
        }}
```

2. In the Helper class, four compilation errors are present. Identify these errors and suggest corrections.

Error	Correction

3. Why is the Member class defined as abstract even though it does not contain any abstract methods?

4. From which class does the toString() method in Helper inherit? Is it an overriding or an overloading? Justify your answer.

5. We want to be able to create an object of the Helper class using a no-argument constructor, as follows: `Helper v = new Helper();`

Is it possible? If not, explain why and suggest a solution.

6. We modify the previous code by removing the boolean isManager parameter from the Helper class constructor and adding the HelperManager subclass to manage other helpers in addition to directly helping recipients. How can we prevent the inheritance of the HelperManager class?