Topic: Enumerations.
OOP concepts involved: Enumerations, Class Instantiation and Declaration, Class Members, Polymorphism, Class Modifiers, Abstract Methods.
Programming generic concepts involved: Access Modifiers, Functions, Primitive Variables, Variable Declaration, Function Parameters.

➢ Theoric introduction

WHAT IS A JAVA ENUM?

A Java Enum (short for Enumeration) is a special Java type used to define collections of constants. More precisely, a Java enum type is a special kind of Java class. An enum can contain constants, methods, etc. *Java enums were added in Java 5*.

Because they are constants, the names of an enum type's fields are in uppercase letters.
In the Java programming language, you define an enum type by using the **enum** keyword. For example, you would specify a days-of-the-week enum type as:

```
public enum Day {
// Java Enum of the days of the week
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
}
```

Notice the **enum** keyword which is used in place of class or interface. The Java enum keyword signals to the Java compiler that this type definition is an enum.

Note: If there is a need for an enum to have a constructor, the constructor for an enum type must be package-private or private access. It automatically creates the constants that are defined at the beginning of the enum body. You cannot invoke an enum constructor yourself.

WHY ARE ENUMS USEFUL?

Enums are used when we know all possible values at **compile time**, such as choices on a menu, rounding modes, command line flags, etc. It is not necessary that the set of constants in an enum type stay fixed for all time.

Java programming language enum types are much more powerful than their counterparts in other languages. The enum declaration defines a class (called an enum type). The enum class body can include methods and other fields. The compiler automatically adds some special methods when it creates an enum. For example, they have a static **values method** that returns an array containing all of the values of the enum in the order they are declared. This method is commonly used in combination with the for-each construct to iterate over the values of an enum type. **For example, this code from the Day enum example below iterates over all the days of the week.**

```
System.out.println("These are the days of the week");
for (Day d: Day.values()) {
       System.out.println(d);
}
```

Note: All enums implicitly extend java.lang.Enum. Because a class can only extend one parent the Java language does not support multiple inheritance of state, and therefore an enum cannot extend anything else.

DECLARATION OF ENUM IN JAVA

You can define an enum type either independently outside of any Class or as part of a Class, but not inside a method.

ENUM DECLARATION INSIDE A CLASS

```
public class Test
{
    enum Color
    {
```

```
        RED, GREEN, BLUE;
    }

    // Driver method
    public static void main(String[] args)
    {
        Color c1 = Color.RED;
        System.out.println(c1);
    }
}
```

ENUM DECLARATION OUTSIDE A CLASS

```
enum Color
{
    RED, GREEN, BLUE;
}

public class Test
{
    // Driver method
    public static void main(String[] args)
    {
        Color c1 = Color.RED;
        System.out.println(c1);
    }
}
```

Note: Java requires that the constants be defined first, prior to any fields or methods. Also, when there are fields and methods, the list of enum constants must end with a semicolon.

➢ Statement

Make an implementation of an enumeration that represents the five most spoken native languages of the world, naming each language as a part of the enum literals (constants). This enum needs to have one private final attributes (nativeSpeakers) related to the language itself. In addition of having attributes and a list of enum literals, implement an abstract class named *speak* that will be overridden in each CommonLanguage constant of the enum.

➢ Class design (UML)

➢ Program Code

CommonLanguage.java

```
public enum CommonLanguage {
	// Declaring enum constants with a value of corresponding to
	// the million of nativeSpekers
	CHINESE(1200){
		@Override
		public void speak() {
			System.out.println("Ni hao");
		}
	},SPANISH(400) {
		@Override
		public void speak() {
```

```
                System.out.println("Hola");
            }
        }, ENGLISH(360) {
            @Override
            public void speak() {
                System.out.println("Hello");
            }
        }, HINDI(322) {
            @Override
            public void speak() {
                System.out.println("Namaste");
            }
        }, JAPANESE(130) {
            @Override
            public void speak() {
                System.out.println("Kon'nichiwa");
            }
        };

        // Constructor Method
        private CommonLanguage(int nativeSpeakers) {
            this.nativeSpeakers = nativeSpeakers;
        }

        // Number of millions of native speakers
        private final int nativeSpeakers;

        // Getter method
        public int getNativeSpeakers() {
            return nativeSpeakers;
        }

        // Abstract method implemented within all constant literals
        public abstract void speak();
}
```

Main.java

```
public class Main {

        public static void main(String[] args) {
            // Using for Each along with the static values() method
            for (CommonLanguage lang : CommonLanguage.values()) {
                System.out.println("There are " + (lang.getNativeSpeakers()) +
                " millions of " + lang + " native speakers, and they speak "
```

```
                    + "like these: ");
                    // Calling the speak method for each CommonLanguage constant
                    lang.speak();
                    System.out.println("");
            }
        }

}
```

➢ Program execution

The CommonLanguage enum has 5 constants, each one of them has a value according to the number of millions of native speakers. Those values are assigned when there is an instantiation of the enum, where the constructor is implicitly called. The value that the constructor receives is stored inside a private final int variable named *nativeSpeakers*.

Just in the Main method, a for-Each is done with the help of the static *values()* method, which belongs to the Java.lang.Enum superclass, from which all enum types inherit. The *values()* method returns an array that contains all the constant values of the enum that we specify.

In each iteration of the for-Each loop, we cover each CommonLanguage constant and have the opportunity to use the own implementation of the *speak()* method along with being able to obtain the specific value that the *nativeSpeakers* constant takes.

```
There are 1200 millions of CHINESE native speakers, and they speak like these:
Ni hao

There are 400 millions of SPANISH native speakers, and they speak like these:
Hola

There are 360 millions of ENGLISH native speakers, and they speak like these:
Hello)

There are 322 millions of HINDI native speakers, and they speak like these:
Namaste

There are 130 millions of JAPANESE native speakers, and they speak like these:
Kon'nichiwa
```

➢ Conclusions

Enumerations are very useful when we need to have a finite collection of predetermined constant literals (values) that we know we are going to use at the compile time.

Java enums can contain constants, attributes, methods, etc. We can add values to the enum constants, having the ability to predefine the value before the constructor is called. Methods and Attributes can be added to the enum class, so that we can perform a series of instructions related to its own enum literals.

And as in the previous example we can use abstract methods within the enum class so that each enum constant can define its own implementation.