

Introduction to Computer Programming (Java A)

Lab 10

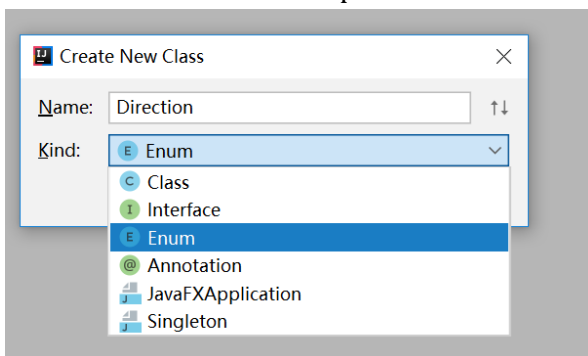
[Objective]

- Learn to use enum types.

Enumerations

An enum type is a special data type that enables a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it. For example, a week has seven days (MONDAY to SUNDAY).

A enum type is declared using the enum keyword, not class. Let's create a new enum type Direction with four constants named "NORTH", "SOUTH", "EAST", and "WEST", respectively. In IDEA, creating a new enum type is similar to creating a new class. The only difference is to select "Enum" in the dropdown list.



```
package sustech.cs102a.lab10;
```

```
public enum Direction {  
    NORTH, SOUTH, EAST, WEST // semicolon unnecessary  
}
```

Variables of this enum type Direction can only receive the values of the four enum constants. For example, the following code creates an object of this enum type.

```
package sustech.cs102a.lab10;
```

```
public class DirectionTest {  
    public static void main(String[] args) {  
        Direction d = Direction.EAST;  
        System.out.println(d);  
    }  
}
```

The above code prints "EAST". The last statement in the main method is equivalent to `System.out.println(d.toString())`. The `toString()` method returns the name of the enum constant EAST.

In the code, **we cannot create an object of the enum type using the "new" operator with a constructor call**. If you compile the following code, you will receive the error message "Enum types cannot be instantiated".

```
public Direction d = new Direction();
```

This is because under the hood, every enum type is internally implemented using class (**the compiler will create a private constructor that cannot be called outside the enum type**).

```
public final class Direction extends Enum {  
    public static final Direction NORTH = new Direction();  
    public static final Direction SOUTH = new Direction();  
    public static final Direction EAST = new Direction();  
    public static final Direction WEST = new Direction();  
} // simplified for illustration
```

From this internal view, we can see that NORTH, SOUTH, EAST, WEST are no more than four class variables pointing to four Direction objects. The final modifier makes them constants.

An enum type variable can be passed as an argument to a “switch” statement.

```
package sustech.cs102a.lab10;  
  
public class DirectionTest {  
    private Direction d;  
  
    public DirectionTest(Direction d) {  
        this.d = d;  
    }  
  
    public Direction getDirection() {  
        return d;  
    }  
  
    public static void main(String[] args) {  
        DirectionTest test = new DirectionTest(Direction.EAST);  
        switch(test.getDirection()) {  
            case EAST: // must be unqualified name of the enum constant  
                System.out.println("Countries in the east: Japan, Korea");  
                break;  
            case WEST:  
                System.out.println("Countries in the west: US, Germany");  
                break;  
            case SOUTH:  
                System.out.println("Countries in the south: Australia, New  
Zealand");  
                break;  
            case NORTH:  
                System.out.println("Countries in the north: Russia, Mongolia");  
                break;  
        }  
    }  
}
```

When declaring an enum type, *besides the enum constants, we can also declare other members* such as **constructors**, **fields** and **methods**. A enum type constructor can specify any number of parameters and can be overloaded, but it cannot have the access modifier public (**must be private or no-modifier, meaning package private**).

```
package sustech.cs102a.lab10;
```

```

public enum Book {
    JHTP("Java: How to Program", "2012"),
    CHTP("C: How to Program"),
    CPPHTP("C++: How to Program", "2012"),
    VBHTP("Visual Basic: How to Program", "2011"),
    CSHARPHTP("Visual C#: How to Program");

    private final String title;
    private final String year;

    private Book(String title, String year) {
        this.title = title;
        this.year = year;
    }

    private Book(String title) {
        this.title = title;
        this.year = "no info";
    }

    public String getTitle() {
        return title;
    }

    public String getYear() {
        return year;
    }
}

```

In the enum type `Book`, there are two fields: `title` and `year`. They are declared to be constants since enum type objects only receive predefined constant values (enum constants). There are two getter methods. There are two overloaded constructors. The two constructors are used in the declarations of the enum constants. For example, when declaring the enum constant `CHTP`, the one-argument constructor is used.

We can further write the following program to test the enum type.

```

package sustech.cs102a.lab10;
import java.util.EnumSet;

public class BookTest {
    public static void main(String[] args) {
        System.out.println("All books:");

        for (Book book : Book.values()) {
            System.out.printf("%-10s", book);
            System.out.printf("%-30s", book.getTitle());
            System.out.printf("%s\n", book.getYear());
        }

        System.out.println("\nDisplaying a range of enum
constants:");

        for (Book book : EnumSet.range(Book.JHTP, Book.CPPHTP)) {
            System.out.printf("%-10s", book);
            System.out.printf("%-30s", book.getTitle());
            System.out.printf("%s\n", book.getYear());
        }
    }
}

```

```

    }
}

```

The code prints:

All books:		
JHTP	Java: How to Program	2012
CHTP	C: How to Program	no info
CPPHTP	C++: How to Program	2012
VBHTP	Visual Basic: How to Program	2011
CSHARPHTP	Visual C#: How to Program	no info
Displaying a range of enum constants:		
JHTP	Java: How to Program	2012
CHTP	C: How to Program	no info
CPPHTP	C++: How to Program	2012

In the above example, only five Book objects will be created. The constants such as Book.JHTP stores the references to the objects.

The **values()** method is a static method that is automatically generated by the compiler to return an array of the enum constants (an array of references to the objects of the enum type).

The generic class EnumSet's static method range() returns a collection of the enum constants in the range specified by two endpoints. In the above code, range() takes two enum constants as arguments. **The first constant should be declared before the second** (the ordinal() method of a enum constant can return the position of the constant in all declared constants). If this constraint is violated (for example, when **EnumSet.range(Book.CPPHTP, Book.JHTP)** is used in the code), an **java.lang.IllegalArgumentException** will be thrown.

Lab exercise:

1. Create an enum type PhoneModel, which contains the following constants: IPHONE, HUAWEI, PIXEL, SAMSUNG, LG.
2. Create a field named price (int type). Write a getter method for this field.
3. Create a one-argument constructor PhoneModel(int price) that can be used to create the enum constants. The prices for the five models are: 9999, 8888, 6666, 9399, 5588.
4. Write a test program .It contains a main method that recommends possible phones for a user based on the user's budget.

Three sample runs:

Your budget: 4000 You do have sufficient money
--

Your budget: 8888 HUAWEI price: 8888 PIXEL price: 6666 LG price: 5588
--

Your budget: 10000 IPHONE price: 9999 HUAWEI price: 8888 PIXEL price: 6666 SAMSUNG price: 9399 LG price: 5588
--