Enums

This is a special data type that allows you to define a list of named constants that behave like objects. For example, you can use an enum to represent the days of the week, months in a year or status codes.

Enums automatically extend the `java.lang.Enum` class, so you can't extend any other class. However, you can implement interfaces in an enum.

Being that the enum is extended you get additional methods such as `name()`, `ordinal()`, `toString()`, `equals()`, and `hashCode()`.

The enum can have constructors, and methods. The constructors and methods can be declared after the constants. If you want to have a custom constructor that takes in a parameter you have to have a value for each constant (enum element).

Here's an example of how you can define an enum in Java:

```
public enum Day {
    MONDAY(1),
    TUESDAY(2),
    WEDNESDAY(3),
    THURSDAY(4),
    FRIDAY(5),
    SATURDAY(6),
    SUNDAY(7);

    int dayOfWeek;


    public boolean Day(int dayOfWeek ) {
        this.dayOfWeek = dayOfWeek;
    }
}
```

With enums you can't make an instance of them, you can only use the constants that are defined in the enum. For example, you can't do this:

```
Day day = new Day();  // Error: Cannot instantiate the type Day
```

Instead, you can use the constants defined in the enum:

```
Day day = Day.MONDAY; // This has the value Day.MONDAY
```

```
public class EnumExample {
    public static void main(String[] args) {
        System.out.println(Day.MONDAY.dayOfWeek);  // Output: 1
    }
}
```

How does an enum work?

When you define an enum, the compiler automatically generates a class that extends the `java.lang.Enum` class.

For example, the `Day` enum we defined above is compiled to the following class:

```
enum Day {
  MONDAY,
  TUESDAY,
  WEDNESDAY,
  THURSDAY,
  FRIDAY,
  SATURDAY,
  SUNDAY
}


// The compiler generates a class that extends the java.lang.Enum class

final class Day extends Enum<Day> {
  public static final Day MONDAY = new Day();
  public static final Day TUESDAY = new Day();
  public static final Day WEDNESDAY = new Day();
  public static final Day THURSDAY = new Day();
  public static final Day FRIDAY = new Day();
  public static final Day SATURDAY = new Day();
  public static final Day SUNDAY = new Day();

  private Day() {
    // Private constructor to prevent external instantiation
  }

  // Additional methods and code for the enum
}
```

The class is declared `final` to prevent it from being extended. The constructor is private to prevent external instantiation. The enum constants are declared `public`, `static`, and `final` to make them accessible as constants.

Enum can implement interfaces

An enum can implement interfaces. For example, you can define an interface called `Food` and have the `Day` enum implement it:

```
interface Food {
  void eat();
}

enum Day implements Food {
  MONDAY,
  TUESDAY,
  WEDNESDAY,
  THURSDAY,
  FRIDAY,
  SATURDAY,
  SUNDAY;

  @Override
  public void eat() {
    System.out.println("This is the eat method.");
  }
}
```