Enum

Enums are lists of constants. When you need a predefined list of values which do not represent some kind of numeric or textual data, you should use an enum. For instance, in a chess game you could represent the different types of pieces as an enum:

enum ChessPiece { PAWN, ROOK, KNIGHT, BISHOP, QUEEN, KING; }

You should always use enums when a variable (especially a method parameter) can only take one out of a small set of possible values. Examples would be things like type constants (contract status: “permanent”, “temp”, “apprentice”), or flags (“execute now”, “defer execution”).

***1)****All* enums implicitly extend java.lang.Enum. Since Java does not support multiple inheritance, an enum cannot extend anything else.

**2)**Enum in Java are type-safe: Enum has there own name-space. It means your enum will have a type for example “Company” in below example and you can not assign any value other than specified in Enum Constants.

Java

|  |  |
| --- | --- |
|  | public enum Company {  EBAY, PAYPAL, GOOGLE, YAHOO, ATT  }  Company cName = Company.EBAY;  cName = 1;   // Compilation Error |

**3)**You can specify values of enum constants at the creation time. MyEnum.values() returns an array of MyEnum’s values.

Java

|  |  |
| --- | --- |
|  | package com.crunchify.tutorial;  /\*\*  \* @author Crunchify.com  \*/    public class CrunchifyEnumExample {    public enum Company {  EBAY(30), PAYPAL(10), GOOGLE(15), YAHOO(20), ATT(25);  private int value;    private Company(int value) {  this.value = value;  }  }    public static void main(String[] args) {  for (Company cName : Company.values()) {  System.out.println("Company Value: " + cName.value + " - Comapny Name: " + cName);  }  }  } |

Output:

|  |  |
| --- | --- |
| 1  2  3  4  5 | Company Value: 30 - Comapny Name: EBAY  Company Value: 10 - Comapny Name: PAYPAL  Company Value: 15 - Comapny Name: GOOGLE  Company Value: 20 - Comapny Name: YAHOO  Company Value: 25 - Comapny Name: ATT |

**4)**Enum constants are implicitly static and final and can not be changed once created.

**5)** Enum can be safely compare using “==” equality operator

Java

|  |  |
| --- | --- |
| 1  2  3  4 | Company eBay = Company.EBAY;  if(eBay == Company.EBAY){    log.info("enum in java can be compared using ==");  } |

**6)** You can not create instance of enums by using new operator in Java because constructor of Enum in Java can only be private and Enums constants can only be created inside Enums itself.

**7)** Instance of Enum in Java is created when any Enum constants are first called or referenced in code.

**8)** An enum specifies a list of constant values assigned to a type.

**9)** An enum can be declared outside or inside a class, but NOT in a method.

**10)** An enum declared outside a class must NOT be marked static, final , abstract, protected , or private

**11)** Enums can contain constructors, methods, variables, and constant class bodies.

**12)** enum constants can send arguments to the enum constructor, using the syntax BIG(8), where the int literal 8 is passed to the enum constructor.

**13)** enum constructors can have arguments, and can be overloaded.

**14)** enum constructors can NEVER be invoked directly in code. They are always called automatically when an enum is initialized.

**15)** The semicolon at the end of an enum declaration is optional. These are legal:

* enum Foo { ONE, TWO, THREE}
* enum Foo { ONE, TWO, THREE};