**Java Enum**

JUNE 28, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [12 COMMENTS](http://www.journaldev.com/716/java-enum#comments)

Java Enum was introduced in Java 1.5 as a new type whose fields consists of fixed set of constants. For example, we can create Direction as Java Enum with fixed fields as EAST, WEST, NORTH, SOUTH.

**Java Enum**

In this tutorial, we will learn know how to create a Java Enum, what are the benefits of using java enum and features of enum types. We will also learn using Java Enum valueOf, enum values, EnumSet and EnumMap with examples.

Java **enum** keyword is used to create an enum type. Let’s have a look at the java enum example program.

package com.journaldev.enums;

public enum ThreadStates {

START,

RUNNING,

WAITING,

DEAD;

}

In above example, ThreadStates is the enum with fixed constants fields START, RUNNING, WAITING and DEAD.

Now let’s see how java enum is better than normal constants fields in java classes.

Let’s create a similar constants class in java.

package com.journaldev.enums;

public class ThreadStatesConstant {

public static final int START = 1;

public static final int WAITING = 2;

public static final int RUNNING = 3;

public static final int DEAD = 4;

}

Now let’s see both enum and constants in usage:

/\*\*

\* This method shows the benefit of using Enum over Constants

\*/

private static void benefitsOfEnumOverConstants() {

//Enum values are fixed

simpleEnumExample(ThreadStates.START);

simpleEnumExample(ThreadStates.WAITING);

simpleEnumExample(ThreadStates.RUNNING);

simpleEnumExample(ThreadStates.DEAD);

simpleEnumExample(null);

simpleConstantsExample(1);

simpleConstantsExample(2);

simpleConstantsExample(3);

simpleConstantsExample(4);

//we can pass any int constant

simpleConstantsExample(5);

}

private static void simpleEnumExample(ThreadStates th) {

if(th == ThreadStates.START) System.out.println("Thread started");

else if (th == ThreadStates.WAITING) System.out.println("Thread is waiting");

else if (th == ThreadStates.RUNNING) System.out.println("Thread is running");

else System.out.println("Thread is dead");

}

private static void simpleConstantsExample(int i) {

if(i == ThreadStatesConstant.START) System.out.println("Thread started");

else if (i == ThreadStatesConstant.WAITING) System.out.println("Thread is waiting");

else if (i == ThreadStatesConstant.RUNNING) System.out.println("Thread is running");

else System.out.println("Thread is dead");

}

If we look at the above example, we have two risks with using constants that are solved by enum.

1. We can pass any int constant to the simpleConstantsExample method but we can pass only fixed values to simpleEnumExample, so it provides type safety.
2. We can change the int constants value in ThreadStatesConstant class but the above program will not throw any exception. Our program might not work as expected but if we change the enum constants, we will get compile time exception which removes any possibility of runtime issues.

Output of the above code snippet is:

Thread started

Thread is waiting

Thread is running

Thread is dead

Thread is dead

Thread started

Thread is waiting

Thread is running

Thread is dead

Thread is dead

Now let’s see more features of java enum with an example:

package com.journaldev.enums;

import java.io.Closeable;

import java.io.IOException;

/\*\*

\* This Enum example shows all the things we can do with Enum types

\*

\*/

public enum ThreadStatesEnum implements Closeable{

START(1){

@Override

public String toString(){

return "START implementation. Priority="+getPriority();

}

@Override

public String getDetail() {

return "START";

}

},

RUNNING(2){

@Override

public String getDetail() {

return "RUNNING";

}

},

WAITING(3){

@Override

public String getDetail() {

return "WAITING";

}

},

DEAD(4){

@Override

public String getDetail() {

return "DEAD";

}

};

private int priority;

public abstract String getDetail();

//Enum constructors should always be private.

private ThreadStatesEnum(int i){

priority = i;

}

//Enum can have methods

public int getPriority(){

return this.priority;

}

public void setPriority(int p){

this.priority = p;

}

//Enum can override functions

@Override

public String toString(){

return "Default ThreadStatesConstructors implementation. Priority="+getPriority();

}

@Override

public void close() throws IOException {

System.out.println("Close of Enum");

}

}

**Java Enum Important Points**

Below are some of the important points with Enum in Java.

1. All java enum implicitly extends **java.lang.Enum** class that extends Object class and implements Serializable and Comparable interfaces. So we can’t extend any class in enum.
2. Since enum is a keyword, we can’t end package name with it, for example com.journaldev.enum is not a valid package name.
3. Java enum can implement interfaces. As in above enum example, it’s implementing Closeable interface.
4. Java enum constructors are always private.
5. We can’t create instance of enum using new operator.
6. We can declare abstract methods in java enum, then all the enum fields must implement the abstract method. In above example getDetail() is the abstract method and all the enum fields have implemented it.
7. We can define a method in enum and enum fields can override them too. For example, toString() method is defined in enum and enum field START has overridden it.
8. Java enum fields has namespace, we can use enum field only with class name like ThreadStates.START
9. Enums can be used in switch statement, we will see it in action in the later part of this tutorial.
10. We can extend existing enum without breaking any existing functionality. For example, we can add a new field NEW in ThreadStates enum without impacting any existing functionality.
11. Since enum fields are constants, java best practice is to write them in block letters and underscore for spaces. For example EAST, WEST, EAST\_DIRECTION etc.
12. Enum constants are implicitly static and final
13. Enum constants are final but it’s variable can still be changed. For example, we can use setPriority() method to change the priority of enum constants. We will see it in usage in below example.
14. Since enum constants are final, we can safely compare them using “==” and equals() methods. Both will have the same result

**Java Enum Example**

Now we know most of the features of Enum, let’s have a look at Java Enum example program. Then we will learn some more features of enum.

package com.journaldev.enums;

import java.io.IOException;

import java.util.EnumMap;

import java.util.EnumSet;

import java.util.Set;

public class JavaEnumExamples {

public static void main(String[] args) throws IOException {

usingEnumMethods();

usingEnumValueOf();

usingEnumValues();

usingEnumInSwitch(ThreadStatesEnum.START);

usingEnumInSwitch(ThreadStatesEnum.DEAD);

usingEnumMap();

usingEnumSet();

}

private static void usingEnumSet() {

EnumSet<ThreadStatesEnum> enumSet = EnumSet.allOf(ThreadStatesEnum.class);

for(ThreadStatesEnum tsenum : enumSet){

System.out.println("Using EnumSet, priority = "+tsenum.getPriority());

}

}

private static void usingEnumMap() {

EnumMap<ThreadStates, String> enumMap = new EnumMap<ThreadStates,String>(ThreadStates.class);

enumMap.put(ThreadStates.START, "Thread is started");

enumMap.put(ThreadStates.RUNNING, "Thread is running");

enumMap.put(ThreadStates.WAITING, "Thread is waiting");

enumMap.put(ThreadStates.DEAD, "Thread is dead");

Set<ThreadStates> keySet = enumMap.keySet();

for(ThreadStates key : keySet){

System.out.println("key="+key.toString()+":: value="+enumMap.get(key));

}

}

private static void usingEnumInSwitch(ThreadStatesEnum th) {

switch (th){

case START:

System.out.println("START thread");

break;

case WAITING:

System.out.println("WAITING thread");

break;

case RUNNING:

System.out.println("RUNNING thread");

break;

case DEAD:

System.out.println("DEAD thread");

}

}

private static void usingEnumValues() {

ThreadStatesEnum[] thArray = ThreadStatesEnum.values();

for(ThreadStatesEnum th : thArray){

System.out.println(th.toString() + "::priority="+th.getPriority());

}

}

private static void usingEnumValueOf() {

ThreadStatesEnum th = Enum.valueOf(ThreadStatesEnum.class, "START");

System.out.println("th priority="+th.getPriority());

}

private static void usingEnumMethods() throws IOException {

ThreadStatesEnum thc = ThreadStatesEnum.DEAD;

System.out.println("priority is:"+thc.getPriority());

thc = ThreadStatesEnum.DEAD;

System.out.println("Using overriden method."+thc.toString());

thc = ThreadStatesEnum.START;

System.out.println("Using overriden method."+thc.toString());

thc.setPriority(10);

System.out.println("Enum Constant variable changed priority value="+thc.getPriority());

thc.close();

}

}

Before explaining other important features of enum, let’s see the output of the above program.

priority is:4

Using overriden method.Default ThreadStatesConstructors implementation. Priority=4

Using overriden method.START implementation. Priority=1

Enum Constant variable changed priority value=10

Close of Enum

th priority=10

START implementation. Priority=10::priority=10

Default ThreadStatesConstructors implementation. Priority=2::priority=2

Default ThreadStatesConstructors implementation. Priority=3::priority=3

Default ThreadStatesConstructors implementation. Priority=4::priority=4

START thread

DEAD thread

key=START:: value=Thread is started

key=RUNNING:: value=Thread is running

key=WAITING:: value=Thread is waiting

key=DEAD:: value=Thread is dead

Using EnumSet, priority = 10

Using EnumSet, priority = 2

Using EnumSet, priority = 3

Using EnumSet, priority = 4

**Java Enum Features Explained**

1. usingEnumMethods() shows how to create an enum object and how we can use it’s methods. It’s also showing use of setPriority(int i) method to change the variable of enum.
2. usingEnumValueOf() shows the usage of **java.util.EnumvalueOf(enumType, name)** through which we can create an enum object from String. It throws IllegalArgumentException if the specified enum type has no constant with the specified name, or the specified class object does not represent an enum type. It also throws NullPointerException if any of the arguments are null.
3. usingEnumValues() method shows the usage of **values()** method that returns an array containing all of the values of the enum in the order they are declared. Note that this method is automatically generated by java compiler for every enum. You won’t find values() implementation in java.util.Enum class.
4. usingEnumInSwitch() method shows how to use enum constants in switch case.
5. usingEnumMap() method shows use of **java.util.EnumMap**, which is introduced in Java 1.5 Collections Framework. EnumMap is Map implementation for use with enum type keys. All of the keys in an enum map must come from a single enum type that is specified, explicitly or implicitly, when the map is created. We can’t use null as key for EnumMap and EnumMap is not synchronized.
6. usingEnumSet() method shows use of **java.util.EnumSet**, which is Set implementation for use with enum types. All of the elements in an enum set must come from a single enum type that is specified, explicitly or implicitly, when the set is created. EnumSet is not synchronized and null elements are not allowed. It also provides some useful methods like copyOf(Collection c), of(E first, E… rest) and complementOf(EnumSet s).

That’s all for java enum example, please share it if you have learned anything new about java enum types.

FILED UNDER: [JAVA](http://www.journaldev.com/dev/java)

**About Pankaj**

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on [**Google Plus**](https://plus.google.com/118104420597648001532/posts?rel=author), [**Facebook**](https://www.facebook.com/journaldev) or [**Twitter**](https://twitter.com/JournalDev). I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on [**Youtube**](https://www.youtube.com/user/JournalDev).

[« Java Properties File – java.util.Properties](http://www.journaldev.com/712/java-properties-file-java-util-properties)

[Java Annotations Example Tutorial »](http://www.journaldev.com/721/java-annotations-example-tutorial)

**Comments**

1. **Shankar Ram says**

[JUNE 28, 2017 AT 5:33 AM](http://www.journaldev.com/716/java-enum#comment-38504)

Very well written

[Reply](http://www.journaldev.com/716/java-enum#comment-38504)

1. **Chandrika says**

[MAY 23, 2017 AT 11:31 PM](http://www.journaldev.com/716/java-enum#comment-38232)

‘We can extend existing enum without breaking any existing functionality. For example, we can add a new field NEW in ThreadStates enum without impacting any existing functionality.’  
Could you please explain this? How will we be adding new Field? Definitely we are not saying to edit the existing class. Right? Then how do we do that?

[Reply](http://www.journaldev.com/716/java-enum#comment-38232)

1. **Bektur Toktosunov says**

[DECEMBER 27, 2015 AT 6:20 AM](http://www.journaldev.com/716/java-enum#comment-33755)

As it is written in API Enum.valueOf() doesn’t throw IllegalStateException.  
It throws IllegalArgumentException “if the specified enum type has no constant with the specified name, or the specified class object does not represent an enum type” instead

[Reply](http://www.journaldev.com/716/java-enum#comment-33755)

* + **Bektur Toktosunov says**

[DECEMBER 28, 2015 AT 10:46 AM](http://www.journaldev.com/716/java-enum#comment-33767)

And thanks for great article! =)

[Reply](http://www.journaldev.com/716/java-enum#comment-33767)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[JUNE 28, 2016 AT 12:04 PM](http://www.journaldev.com/716/java-enum#comment-35473)

Thanks for catching typo error, i have corrected it.

[Reply](http://www.journaldev.com/716/java-enum#comment-35473)

1. **Shambhu says**

[SEPTEMBER 19, 2015 AT 10:37 AM](http://www.journaldev.com/716/java-enum#comment-33200)

Well written. Nice article Pankaj.

[Reply](http://www.journaldev.com/716/java-enum#comment-33200)

1. **Manish kumar says**

[MAY 19, 2014 AT 5:34 AM](http://www.journaldev.com/716/java-enum#comment-28921)

is it necessary that enum constructor should be private .

[Reply](http://www.journaldev.com/716/java-enum#comment-28921)

1. **raj says**

[JANUARY 7, 2014 AT 1:53 PM](http://www.journaldev.com/716/java-enum#comment-27923)

Enum implicitly extends **java.lang.Enum implements Comparable, Serializable**, but not java.util.Enum which is mention most important points of Enum. Please make a not about that.

[Reply](http://www.journaldev.com/716/java-enum#comment-27923)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[JANUARY 7, 2014 AT 5:41 PM](http://www.journaldev.com/716/java-enum#comment-27924)

Thanks for pointing out the type error, corrected it.

[Reply](http://www.journaldev.com/716/java-enum#comment-27924)

* + **Surya says**

[JANUARY 10, 2015 AT 10:29 AM](http://www.journaldev.com/716/java-enum#comment-31402)

thanks

[Reply](http://www.journaldev.com/716/java-enum#comment-31402)

1. **anonymous says**

[AUGUST 27, 2013 AT 10:05 AM](http://www.journaldev.com/716/java-enum#comment-22775)

@Pankaj Nice Article.

One doubt. Assume if you’re getting an int, how would you map it to appropriate enum type.

void someMethod(int i)  
{  
//how to map i to ThreadStatesEnum type?  
switch (th){  
case START:  
System.out.println("START thread");  
break;  
case WAITING:  
System.out.println("WAITING thread");  
break;  
case RUNNING:  
System.out.println("RUNNING thread");  
break;  
case DEAD:  
System.out.println("DEAD thread");  
}  
}

Regads.

[Reply](http://www.journaldev.com/716/java-enum#comment-22775)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[AUGUST 27, 2013 AT 7:38 PM](http://www.journaldev.com/716/java-enum#comment-22800)

Here is a sample program for you where I am using switch case to map int to Enum.

package com.journaldev.enums;

public class Test {

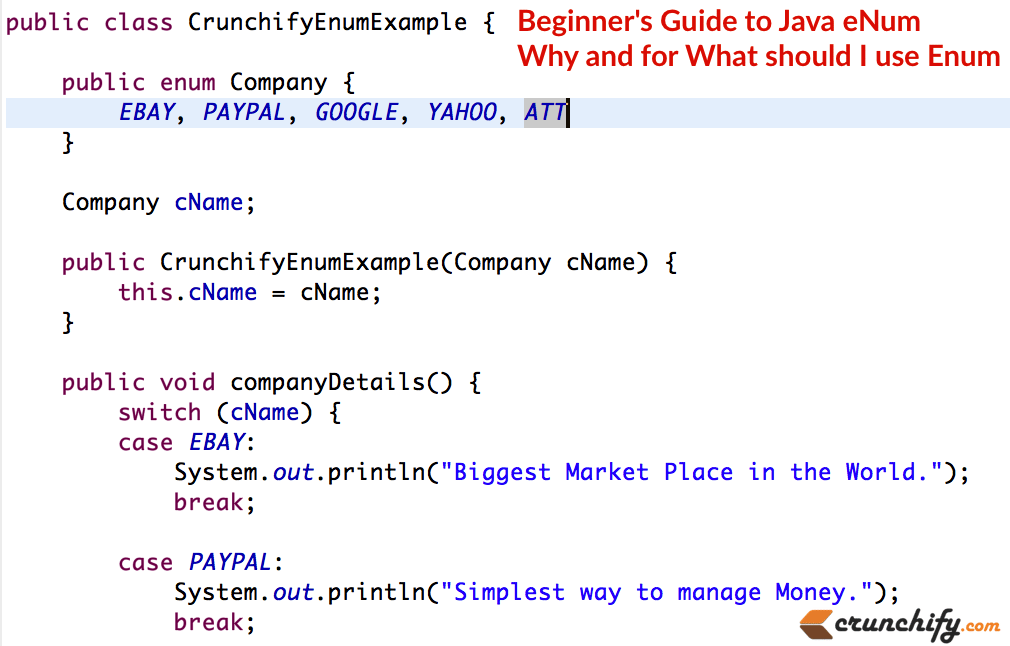
public static void main(String[] args) {  
try {  
someMethod(4);  
} catch (Exception e) {  
e.printStackTrace();  
}  
}

static void someMethod(int i) throws Exception {  
ThreadStatesEnum th = null;

switch(i){  
case 1:  
th = ThreadStatesEnum.START;  
break;  
case 2:  
th = ThreadStatesEnum.RUNNING;  
break;  
case 3:  
th = ThreadStatesEnum.WAITING;  
break;  
case 4:  
th = ThreadStatesEnum.DEAD;  
break;  
}

if(th == null)  
throw new Exception("invalid argument");

//now we can use th in switch statements  
}  
}

[](http://cdn.crunchify.com/wp-content/uploads/2012/11/Beginners-Guide-to-Java-eNum-Why-and-for-What-should-I-use-Enum.png)

Enums are lists of constants. When you need a predefined list of values which do represent some kind of numeric or textual data, you should use an enum. For [instance](http://crunchify.com/lazy-creation-of-singleton-threadsafe-instance-without-using-synchronized-keyword/), 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](http://crunchify.com/what-is-an-interface-in-java-how-its-used-java-tutorial-example-attached/) (especially a method parameter) can only take one out of a small set of possible values. Examples would be things like type [constants](http://crunchify.com/fundamentals-of-java-static-methods-and-variables/) (contract status: “permanent”, “temp”, “apprentice”), or flags (“execute now”, “defer execution”).

If you use enums instead of [integers](http://crunchify.com/what-are-the-difference-between-jdbcs-statement-preparedstatement-and-callablestatement/) (or String codes), you increase [compile](http://crunchify.com/spring-framework-4-order-annotation-tutorial-sort-order-for-an-annotated-component/)-time checking and avoid [errors](http://crunchify.com/have-you-noticed-nullpointerexception-npe-there-are-few-best-practices-to-avoid-npe/) from passing in invalid constants, and you document which values are legal to use.

### ****Java**** Example:

CrunchifyEnumExample.java

Java

|  |
| --- |
| package com.crunchify.tutorials;    /\*\*  \* @author Crunchify.com  \*/    public class CrunchifyEnumExample {    public enum Company {  EBAY, PAYPAL, GOOGLE, YAHOO, ATT  }    Company cName;    public CrunchifyEnumExample(Company cName) {  this.cName = cName;  }    public void companyDetails() {  switch (cName) {  case EBAY:  System.out.println("Biggest Market Place in the World.");  break;    case PAYPAL:  System.out.println("Simplest way to manage Money.");  break;    case GOOGLE:  case YAHOO:  System.out.println("1st Web 2.0 Company.");  break;    default:  System.out.println("Google - biggest search giant.. ATT - my carrier provider..");  break;  }  }    public static void main(String[] args) {  CrunchifyEnumExample ebay = new CrunchifyEnumExample(Company.EBAY);  ebay.companyDetails();  CrunchifyEnumExample paypal = new CrunchifyEnumExample(Company.PAYPAL);  paypal.companyDetails();  CrunchifyEnumExample google = new CrunchifyEnumExample(Company.GOOGLE);  google.companyDetails();  CrunchifyEnumExample yahoo = new CrunchifyEnumExample(Company.YAHOO);  yahoo.companyDetails();  CrunchifyEnumExample att = new CrunchifyEnumExample(Company.ATT);  att.companyDetails();  }  } |

### Output:

Java

|  |
| --- |
| Biggest Market Place in the World.  Simplest way to manage Money.  1st Web 2.0 Company.  1st Web 2.0 Company.  Google - biggest search giant.. ATT - my carrier provider.. |

### Some very important points on Java Enum:

### Point-1

All enums implicitly extend java.lang.Enum. Since Java does not support multiple [inheritance](http://crunchify.com/understanding-java-annotation-annotation-examples/), an enum cannot extend anything else.

### Point-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](http://crunchify.com/java-nio-non-blocking-io-with-server-client-example-java-nio-bytebuffer-and-channels-selector-java-nio-vs-io/) 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 |

### Point-3

You can specify values of enum constants at the creation time. MyEnum.values() returns an [array](http://crunchify.com/java-two-ways-to-convert-char-array-to-string/) 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 + " - Company Name: " + cName);  }  }  } |

Output:

|  |
| --- |
| Company Value: 30 - Company Name: EBAY  Company Value: 10 - Company Name: PAYPAL  Company Value: 15 - Company Name: GOOGLE  Company Value: 20 - Company Name: YAHOO  Company Value: 25 - Company Name: ATT |

### Point-4

Enum constants are implicitly static and [final](http://crunchify.com/in-java-how-to-perform-file-search-operation-using-java-nio-file-interface-tutorial-on-file-and-directory-operations/) and can not be changed once created.

### Point-5

Enum can be safely compare using:

1. Switch-Case Statement
2. == Operator
3. [.equals()](http://crunchify.com/how-to-override-equals-and-hashcode-method-in-java/) method

Java

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

Please [follow complete tutorial](http://crunchify.com/java-enum-comparison-using-equals-operator-switch-case-statement-and-equals-method-complete-example/).

### Point-6

You can not create instance of enums by using new operator in Java because constructor of Enum in [Java](http://crunchify.com/category/java-tutorials/) can only be private and Enums constants can only be created inside Enums itself.

### Point-7

Instance of Enum in Java is created when any Enum constants are first called or [referenced](http://crunchify.com/missing-maven-settings-xml-file-for-your-eclipse-what-if-you-need-two-settings-xml-file-for-work-personal-workspace/) in code.

### Point-8

An enum specifies a list of constant values assigned to a [type](http://crunchify.com/java-tip-wherever-possible-try-to-use-primitive-types-instead-of-wrapper-classes/).

### Point-9

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

### Point-10

An enum declared outside a class must NOT be marked static, final , [abstract](http://crunchify.com/what-is-an-abstract-class-and-abstract-method-in-java-when-should-i-use-it/), protected , or private

### Point-11

Enums can contain [constructors](http://crunchify.com/create-simple-pojo-and-multiple-java-reflection-examples/), methods, variables, and constant class bodies.

### Point-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.

### Point-13

enum constructors can have arguments, and can be [overloaded](http://crunchify.com/java-method-overriding-examples-and-concepts-overriding-rules/).

### Point-14

enum constructors can NEVER be invoked directly in code. They are always called [automatically](http://crunchify.com/how-to-use-expiringmap-maven-java-utility-to-remove-expired-objects-from-map-automatically-complete-java-tutorial/) when an enum is initialized.

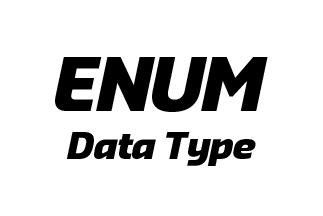
### Point-15

The semicolon at the end of an enum declaration is optional.

**These are legal:**

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

### Another simple Java eNUM Example:

[](http://cdn.crunchify.com/wp-content/uploads/2013/04/enum_data_type-Crunchifiy.png)

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.

You should always use enums when a [variable](http://crunchify.com/fundamentals-of-java-static-methods-and-variables/) (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”).

If you use enums instead of integers (or String codes), you increase compile-time checking and avoid errors from passing in invalid [constants](http://crunchify.com/java-how-to-find-unique-values-in-arraylist-using-treeset-hashset/), and you document which values are legal to use.

Between, overuse of enums might mean that your methods do too much (it’s often better to have several separate methods, rather than one method that takes several flags which modify what it does), but if you have to use flags or type codes, enums are the way to go.

### This is very simple Java eNum Example

CrunchifyEnumCompany.java

|  |
| --- |
| package com.crunchify.tutorials;    /\*\*  \* @author Crunchify.com  \*/    public enum CrunchifyEnumCompany {    GOOGLE("G"), YAHOO("Y"), EBAY("E"), PAYPAL("P");    private String companyLetter;    private CrunchifyEnumCompany(String s) {  companyLetter = s;  }    public String getCompanyLetter() {  return companyLetter;  }  } |

CrunchifyEnumExample.java

|  |
| --- |
| package com.crunchify.tutorials;    import com.crunchify.tutorials.CrunchifyEnumCompany;    /\*\*  \* @author Crunchify.com  \*/    public class CrunchifyEnumExample {    public static void main(String[] args) {  System.out.println("Get enum value for Comapny 'eBay': "  + CrunchifyEnumCompany.EBAY.getCompanyLetter());  }    } |

Output:

|  |
| --- |
| Get enum value for Comapny 'eBay': Value: E |

#### Join the Discussion

Share & leave us some comments on what you think about this topic or if you like to add something.