1. Suppose I have an enum defined say Subjects with Values: DBMS, DSA, NETWORKING, OPERATING SYSTEMS. a) How do I access the index of the values present in the enums. b) If while accessing the enums I pass an argument or say value which is not present in the enum then what kind of exception will be thrown? c) Can you demonstrate the same using a java program?

**Answer:**

1. We can access the index of the method by using pre-defined method namely **ordinal()** which enables us to access index of enums . For that we need to store all the enum values in the array of type enum\_name and then we can access it. Following is the code

**code:**

package com.shahNiket;

enum Subjects {

DBMS, DSA, NETWORKING, OPERATING\_SYSTEMS;

}

public class Books {

public static void main(String[] args) {

Subjects[] s = Subjects.*values*();

for (Subjects subjects : s) {

System.***out***.println("Value of enum " + subjects.name() + " Index of value " + subjects.ordinal());

}

}

} **OUTPUT:**

Value of enum DBMS Index of value 0

Value of enum DSA Index of value 1

Value of enum NETWORKING Index of value 2

Value of enum OPERATING\_SYSTEMS Index of value 3

1. If we try to access the value that is not present in the enum then it will throw **java.lang.IllegalArgumentException: No enum constant.**

Following is the code:

**Code:**

package com.shahNiket;

import java.util.Scanner;

enum Subjects2 {

DBMS, DSA, NETWORKING, OPERATING\_SYSTEMS;

}

public class Books2 {

static void bookUpdate(Subjects2 s) {

switch(s){

case DBMS:System.out.println("Database Management System by Mcc Graw Hill");

break;

case DSA:System.out.println("Data Structures and Algorithms by Thomas H Cormen");

break;

case NETWORKING:System.out.println("Networking by Technical");

break;

case OPERATING\_SYSTEMS:System.out.println("Operating system by Silberschatz, Galvin and Gagne");

break;

}

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String subject=sc.next();

Subjects2 s1=Subjects2.valueOf(subject);

bookUpdate(s1);

}

}

**OUTPUT:**

JAVA

Exception in thread "main" java.lang.IllegalArgumentException: No enum constant com.shahNiket.Subjects2.JAVA

at java.base/java.lang.Enum.valueOf(Enum.java:273)

at Enums/com.shahNiket.Subjects2.valueOf(Books2.java:1)

at Enums/com.shahNiket.Books2.main(Books2.java:25)

DSA

Data Structures and Algorithms by Thomas H Cormen

**2 .**  Will the following method compile? If not, why?

|  |
| --- |
| public static void print(List<? extends Number> list) {  for (Number n : list)  System.out.print(n + " ");  System.out.println();  } |

**Answer:**

Yes, this method compiles successfully. It is **upperbounded wildcard**. So the above method can Only accept the list whose type extends the Number class such as Double, Float, Integer etc. For other type It will give you a compile time Error.

**3.** If the compiler erases all type parameters at compile time, why should you use generics?

**Answer:**

We use Generics because of the following advantages:

* Type safety: Generics make the error to appear at compile time.
* Individual typecasting is not required: Using generics we don’t need to typecast Every time we retrieve data from ArrayList.
* Generics promote code reusability: with the help of generics we can write the code that will work with different types of data.
* Implementing generics Algorithm: By using generics, we can implement the algorithms that work on different kind off objects and they are also type safe.

**4.** What is the following class converted to after type erasure?

public class Pair <K, V> {

public Pair(K key, V value) {

this.key = key;

this.value = value;

}

public K getKey() { return key; }

public V getValue() { return value; }

public void setKey(K key) { this.key = key; }

public void setValue(V value) { this.value = value; }

private K key;

private V value;

}

**Answer:**

After,type erasure the following class is like:

public class Pair {

public Pair(Object key, Object value) {

this.key = key;

this.value = value;

}

public Object getKey() { return key; }

public Object getValue() { return value; }

public void setKey(Object key) { this.key = key; }

public void setValue(Object value) { this.value = value; }

private Object key;

private Object value;

**}**

In type Erasure process the compiler erases all types and replaces each with first bound if type parameter is bounded or Object if type parameter is not bounded.

**5.** Look at the following code snippet and select the correct option:

class Test extends Exception { }

class Main {

public static void main(String args[]) {

try {

throw new Test();

} catch(Test t) {

System.out.println("Got the Test Exception");

}

finally {

System.out.println("Inside finally block ");

}

}

}

Options:

a) Got the Test Exception

Inside finally block

b) Got the Test Exception

c) Inside finally block

d) Compiler Error

**Answer:**

a) Got the Test Exception

Inside finally block

**6.** What will be the output of the following code:

class Test {

public static void main(String[] args) {

try {

int a[]= {1, 2, 3, 4};

for (int i = 1; i <= 4; i++) {

System.out.println ("a[" + i + "]=" + a[i] + "\n");

}

} catch (Exception e) {

System.out.println ("error = " + e);

}

catch (ArrayIndexOutOfBoundsException e) {

System.out.println ("ArrayIndexOutOfBoundsException");

}

}

}

a) Compiler error

b) Run time error

c) ArrayIndexOutOfBoundsException

d) Error Code is printed

e) Array is printed

**Answer:**

a) Compiler error: It is because All catch blocks must be ordered from most specific to most general. So it should be ArrayIndexOutOfBoundsException first then followed by Exception catch block. Then it will compile Successfully.