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 :**

enum Subjects {

DBMS, DSA, NETWORKING, OPERATING\_SYSTEMS;

}

public class book {

public static void main(String[] args){

Subjects[] s = Subjects.values();

for (Subjects subjects : s){

System.out.println("Value of Enum "+subjects.name());

System.out.println("Index Value of Enum "+subjects.ordinal());

}

}

}

**Output:**

Value of Enum DBMS

Index Value of Enum 0

Value of Enum DSA

Index Value of Enum 1

Value of Enum NETWORKING

Index Value of Enum 2

Value of Enum OPERATING\_SYSTEMS

Index Value of Enum 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.**

**Code :**

import java.util.Scanner;

enum Subjects2 {

DBMS, DSA, NETWORKING, OPERATING\_SYSTEMS;

}

public class subject {

static void sub(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) {

System.out.println("Enter Subject");

Scanner sc=new Scanner(System.in);

String subject=sc.next();

sub(Subjects2.valueOf(subject));

}

}

**Output:**

**Invalid Input :**

Enter Subject

dbms

Exception in thread "main" java.lang.IllegalArgumentException: No enum constant Subjects2.dbms

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

at Subjects2.valueOf(subject.java:3)

at subject.main(subject.java:28)

**Valid Input :**

Enter Subject

DBMS

Database Management System by Mcc Graw Hill

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

public static void print(List list)

{

for (Number n : list)

System.out.print(n + " ");

System.out.println();

}

**Answer:**

Yes, this method is upperbound wildcard.

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

**Answer:**

Generics promote code reusability

Individual typecasting is not required

Generics support programming types as parameters.

**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:**

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;

**}**

**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 ");

}

}

}

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