**Question Bank**

**Module 3**

**Object Oriented Programming using Java (21CIC34)**

**Semester:3rd Session : July-Dec 2022**

1. What is string?Explain the two ways of creating String object. Illustrate how the string object created using String and StringBuffer class constructors.( CO3)

### What is String in Java?

Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The java.lang.String class is used to create a string object.

### How to create a string object?

There are two ways to create String object:

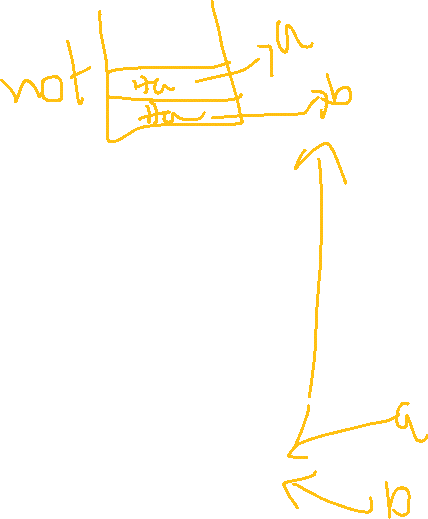
1. By string literal
2. By new keyword

### 1) String Literal

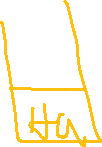
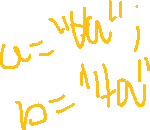
Java String literal is created by using double quotes. For Example:

1. String s="welcome";

Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, then the new var points to the same reference



1. String s1="Welcome";
2. String s2="Welcome";//It doesn't create a new instance







In the above example, only one object will be created. Firstly, JVM will not find any string object with the value "Welcome" in string constant pool that is why it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.



### 2) By new keyword

1. String s=**new** String("Welcome");//creates two objects and one reference variable

In such case, [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

### Java String Example

**StringExample.java**

1. **public** **class** StringExample{
2. **public** **static** **void** main(String args[]){
3. String s1="java";//creating string by Java string literal
4. **char** ch[]={'s','t','r','i','n','g','s'};
5. String s2=**new** String(ch);//converting char array to string
6. String s3=**new** String("example");//creating Java string by new keyword
7. System.out.println(s1);
8. System.out.println(s2);
9. System.out.println(s3);
10. }}

[**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=StringExample)

**Output:**

java

strings

example

The above code, converts a **char** array into a **String** object. And displays the String objects **s1, s2**, and **s3** on console using **println()** method.

Illustrate how the string object created using String class constructors.

|  |
| --- |
| class stingcons  {      public static void main(String[] args)      {          //empty string          String x=new String();          System.out.println(x);          //string created using new keyword          String x1=new String("hasni");          System.out.println(x1);          char a[]={'a','b','c','d','e'};          //char arr to string          String c=new String(a);          System.out.println(c);          //subrange of char arr to string          String c1=new String(a,1,3);//dtring(char arr,start index,noofchars)          System.out.println(c1);          //st{ring created using array of byte cahr          byte b1[]={64,65,66,67,68,69};          String bs1=new String(b1);          String bs2=new String(b1,1,3);//(byte array,startindex,endindex)          System.out.println("string of array of byte"+bs1);          System.out.println("string of subrange of array of byte "+bs2);      }    } |

output

|  |
| --- |
| hasni  abcde  bcd  string of array of byte@ABCDE  string of subrange of array of byte ABC |

Illustrate how the string object created using String class constructors

public class stringfromstrigbuffer {

  public static void main(String[] args) {

    StringBuffer x=new StringBuffer("hansi");

    String x1=new String(x);

    System.out.println("string buffer: "+x);

    System.out.println("string from string buffer: "+x1);

  }

}



1. Distinguish between String and StringBuffer class.( CO3)

|  |  |  |
| --- | --- | --- |
| **No.** | **String** | **StringBuffer** |
| 1) | The String class is immutable. | The StringBuffer class is mutable. |
| 2) | String is slow and consumes more memory when we concatenate too many strings because every time it creates new instance. | StringBuffer is fast and consumes less memory when we concatenate. bcz no new instance is created |
| 3) | String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method to check whether both strings have same characters in same sequence. | StringBuffer class doesn't override the equals() method of Object class. |
| 4) | String class is slower while performing concatenation operation. | StringBuffer class is faster while performing concatenation(by append or other methods) operation. |
| 5)  6) | String class uses String constant pool.  There wont be anything like capacity in strings as they are immutable | StringBuffer uses Heap memory  stringbuffer will be having something called as capacity() |

1. Demonstrate the different methods used in String class with an example.( CO3)

1)String length:

The length of a string is the number of characters that it contains.

|  |
| --- |
| * + int length( )   + char chars[] = { 'a', 'b', 'c' };     - String s = new String(chars);     - System.out.println(s.length());     - //output: 3 |

ADDITITIONAL:

System.out.println("abc".length()); // Valid Function on string Literal Directly.

* BCZ: String s2 = "abc";
* Java automatically constructs a String object. Thus, you can use a string literal to initialize a String object.
* String object is created for every string literal, you can use a string literal any place you can use a String object

2) Character extraction:

* The String class provides a number of ways in which characters can be extracted from a String object.
* **charAt( )**
  + To extract a single character from a String, you can refer directly to an individual character via the charAt( ) method. It has this general form:
  + char charAt(int *where)*
  + *Where, is the index of the character that you want to obtain. The value of where must be* nonnegative and specify a location within the string.
  + charAt( ) returns the character at the specified location.

For example

|  |
| --- |
| String h=”hasni”;  System.out.println(h.charAt(1));  Output:  a |

1)equals() method

The equals() method compares two strings, and returns true if the strings are equal, and false if not.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  String myStr1 = "Hello";  String myStr2 = "Hello";  String myStr3 = "Another String";  System.out.println(myStr1.equals(myStr2));  System.out.println(myStr1.equals(myStr3));  }  } |

4)concat method

The concat() method appends (concatenate) a string to the end of another string.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  String firstName = "John ";  String lastName = "Doe";  System.out.println(firstName.concat(lastName));  }  } |

Output:

John Doe

5)indexof

The indexOf() method returns the position of the first occurrence of specified character(s) in a string.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  String myStr = "kkt";  System.out.println(myStr.indexOf("t")); //note t in invited commass  }  } |

6)lastIndexOf()

The lastIndexOf() method returns the position of the last occurrence of specified character(s) in a string.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  String myStr = "hasnintn";  System.out.println(myStr.lastIndexOf("n"));  }  }  Output  7 |

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Return Type** |
| [charAt()](https://www.w3schools.com/java/ref_string_charat.asp) | Returns the character at the specified index (position) | char |
| [codePointAt()](https://www.w3schools.com/java/ref_string_codepointat.asp) | Returns the Unicode of the character at the specified index | int |
| [codePointBefore()](https://www.w3schools.com/java/ref_string_codepointbefore.asp) | Returns the Unicode of the character before the specified index | int |
| [codePointCount()](https://www.w3schools.com/java/ref_string_codepointcount.asp) | Returns the number of Unicode values found in a string.  Note:includes startpoint excludes endpoint | int |
| [compareTo()](https://www.w3schools.com/java/ref_string_compareto.asp) | Compares two strings lexicographically(alphabetically) | int |
| [compareToIgnoreCase()](https://www.w3schools.com/java/ref_string_comparetoignorecase.asp) | Compares two strings lexicographically, ignoring case differences | int |
| [concat()](https://www.w3schools.com/java/ref_string_concat.asp) | Appends a string to the end of another string | String |
| [contains()](https://www.w3schools.com/java/ref_string_contains.asp) | Checks whether a string contains a sequence of characters | boolean |
| [contentEquals()](https://www.w3schools.com/java/ref_string_contentequals.asp) | Checks whether a string contains the exact same sequence of characters of the specified CharSequence or StringBuffer | boolean |
| [equals()](https://www.w3schools.com/java/ref_string_equals.asp) | Compares two strings. Returns true if the strings are equal, and false if not// | boolean |
| [endsWith()](https://www.w3schools.com/java/ref_string_endswith.asp) | Checks whether a string ends with the specified character(s) | boolean |
|  |  | boolean |
| [equalsIgnoreCase()](https://www.w3schools.com/java/ref_string_equalsignorecase.asp) | Compares two strings, ignoring case considerations | boolean |
| format() | Returns a formatted string using the specified locale, format string, and arguments | String |
| getBytes() | Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array | byte[] |
| getChars() | Copies characters from a string to an array of chars | void |
| [hashCode()](https://www.w3schools.com/java/ref_string_hashcode.asp) | Returns the hash code of a string | int |
| [indexOf()](https://www.w3schools.com/java/ref_string_indexof.asp) | Returns the position of the first found occurrence of specified characters in a string | int |
| intern() | Returns the canonical representation for the string object | String |
| [isEmpty()](https://www.w3schools.com/java/ref_string_isempty.asp) | Checks whether a string is empty or not | boolean |
| [lastIndexOf()](https://www.w3schools.com/java/ref_string_lastindexof.asp) | Returns the position of the last found occurrence of specified characters in a string | int |
| [length()](https://www.w3schools.com/java/ref_string_length.asp) | Returns the length of a specified string | int |

1. Demonstrate the methods used in StringBuffer class for inserting,deleting and appending operations.( CO3)

(A)**Insert() method :-** The insert() method inserts the given string with this string at the given position. This method used for inserting operation

import java.io.\* ;

**class A {**

**public static void main(String args[]) {**

**StringBuffer sb = new StringBuffer("Hello ");**

**sb.insert(1, "Java");**

**// Now original string is changed System.out.println(sb);**

**} }**

(B)**Delete() method :-** The delete() method of StringBuffer class deletes the string from the specified beginIndex and endIndex-1.

**import java.io.\* ;**

**class A{**

**public static void main(String args[]){**

**StringBuffer sb=new StringBuffer("Hello"); sb.delete(1,3);**

**System.out.println(sb); }**

**}**

**© append() method :-** The append() method concatenates the given argument with this string

**import java.io.\* ;**

**class A {**

**public static void main(String args[])**

**{**

**StringBuffer sb = new StringBuffer("Hello ");**

**sb.append(“hasni”);**

**// Now original string is changed**

**System.out.println(sb);**

**}**

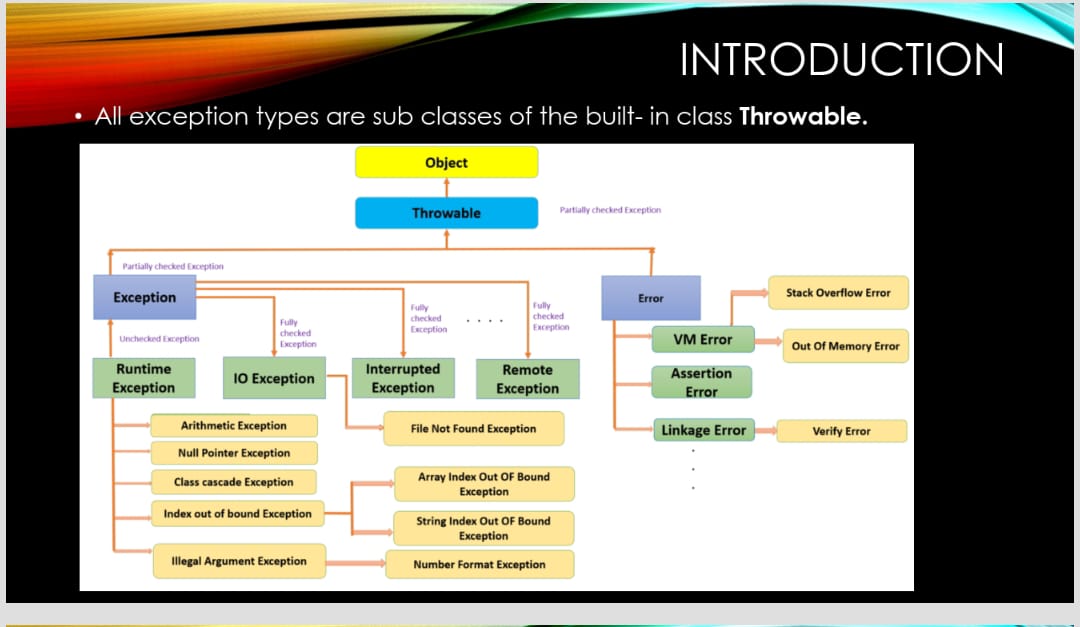
**}**

|  |
| --- |
| public class stringbuffer  {      public static void main(String[] args) {          StringBuffer x=new StringBuffer("hasni");          x.insert(2,"vvv");          System.out.println(x);          //to delete only one ele (indexoftobedeletet,indexoftobedeletet+1)          x.delete(0,1);          System.out.println(x);          x.append("last");          System.out.println(x);      }  } |

Output:

|  |
| --- |
| havvvsni  avvvsni  avvvsnilast |

1. What is an exception? Explain java’s exception hierarchy and the mechanism of exception handling with an example. ( CO3)



**Answer :-** An exception (or exceptional event) is a problem that arises during the execution of a program. Exception leads to abnormal termination of the program

**Types of Exceptions**

There are three categories of Exceptions. They are 1. Checked Exception

2. Unchecked Exception

3. Error

**1.Checked exceptions** − A checked exception is an exception that is checked (notified) by the compiler at compilation-time, these are also called as compile time exceptions.

 These exceptions cannot simply be ignored, the programmer should take care of (handle) these exceptions.

 E.g. IOException, InterruptedException,RemoteException

**2.Unchecked exceptions** − An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions

 These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation.

 E.g. ArithmeticException, IndeOutOfBoundsException, NullPointerException,ClassCascadeException,IllegalArgumentExceptions

**3.Errors** − These are not exceptions at all, but problems that arise beyond the control of the user or the programmer.

 They are also ignored at the time of compilation. Error is irrecoverable.  E.g. VirtualMachineError, AssertionError. OutOfMemoryError,LinkageError etc.

Working of try and catch Block :- Syntax

try {

} catch (Exception e) {

// TODO: handle exception }

The code which is prone to exceptions is placed in the try block. When an exception occurs, that exception occurred is handled by catch block associated with it.

Every try block should be immediately followed either by a catch block or finally block.

A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the catch block (or blocks) that follows the try is checked. If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block.

Example :-

{

    public static void main(String[] args)

    {

        try

        {

            System.out.println(1/0);//AS THIS exception is in try block so this will be handled by catch block with the exception class of this type

            System.out.println("kkr");

        }

        catch(ArithmeticException e)

        {

            System.out.println("division cannot be done by zero");

        }

    }

}

}

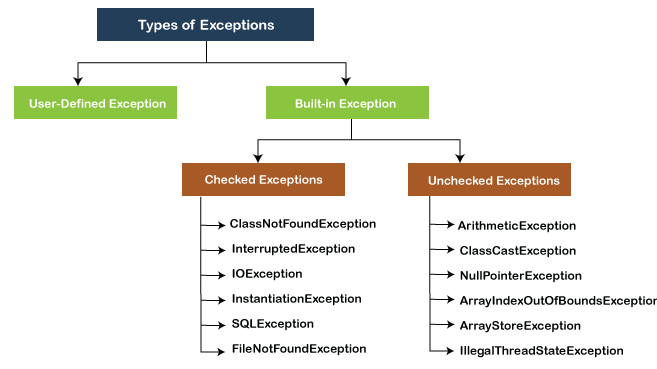
Output:

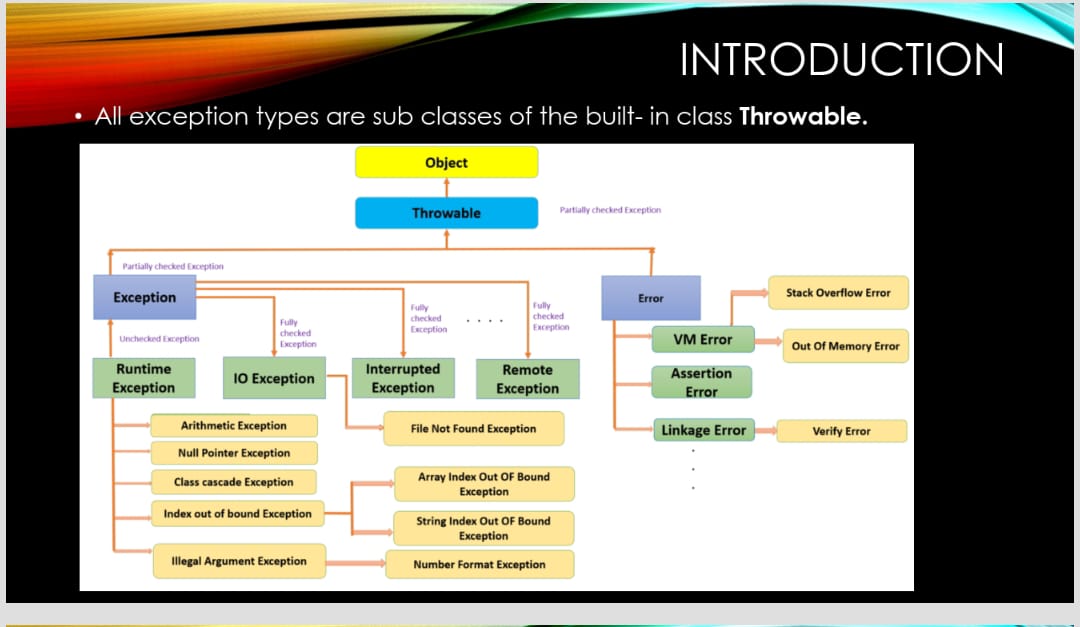
division cannot be done by zero

1. Define Exception.Explain the different types of exception. Demonstrate the working of try and catch statement.( CO3)

Same as above question

1. Mention some of java’s checked exceptions and unchecked exceptions. ( CO3)





Handling InterruptedException

|  |
| --- |
| //thread.sleep might through an interrupted exception  public class hadlingcheckedexception  {      public static void main(String[] args) throws InterruptedException      {          Thread.sleep(10000);      }  } |

1. Demonstrate the working of nested try blocks with example. ( CO3)

In the nested tryblock if any exception is caught in nested try(not outside try) and there are no catch statements of spefic exception type w.r.t. that nested try then the compiler checks whether the catch block is declared with exception to be caught as exception that is raised if yes thacatch block will be executeds

//nested try

class EH4

{

    public static void main(String[] args)

    {

        try{

            try

            {

                System.out.println(1/0);

                int a[]={1,2,3};

                System.out.println(a[5]);

            }

            catch(ArrayIndexOutOfBoundsException e)

            {

                System.out.println("AIOB from inner catch block");

            }

            catch(NullPointerException eNullPointerException)

            {

                System.out.println("error from Exception");

            }

        }

        catch(ArithmeticException e)

        {

            System.out.println("AE from outer catch block");

        }

    }

}

1. Demonstrate the working multiple catch statements with example. ( CO3)

**Answer :-** A try block can be followed by multiple catch blocks. The syntax for multiple catch blocks looks like the following

**Syntax :-**try {

} catch (Exception e) {

// TODO: handle exception }

catch (Exception e) {

// TODO: handle exception

**Example :-**

public class mmmm {

public static void main(String[] args) {

try{

int a[]=new int[5]; a[5]=30/0;

}

catch(ArithmeticException e) {

System.out.println("Arithmetic Exception occurs"); }

catch(ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBounds Exception occurs");

} catch(Exception e)

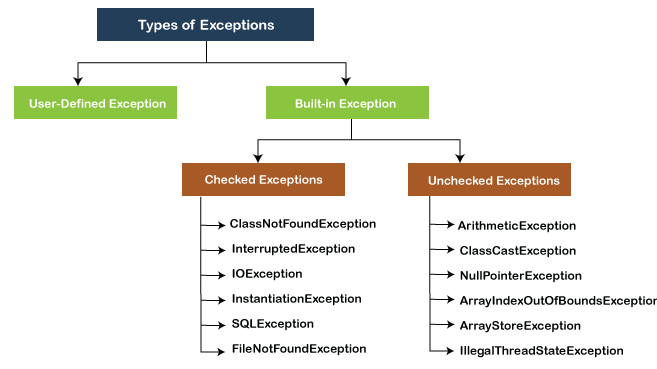
{

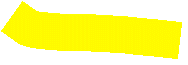
System.out.println("Parent Exception occurs"); }

System.out.println("rest of the code"); }

}

1. Explain the java’s built in exceptions. ( CO3)





1. Create a try block that is likely to generate three types of exception and incorporate necessary catch blocks to catch and handle them. ( CO3)

Try with three exception say

**[NullPointerException:](https://www.geeksforgeeks.org/null-pointer-exception-in-java/)**This exception occurs when trying to access the object reference that has a null value.

* Java

|  |
| --- |
| // Importing Classes/Files  **import** java.io.\*;    **public** **class** GFG {      // Main Driver Method  **public** **static** **void** main(String[] args)      {            // Instance of string a has null value          String a = **null**;            // Comparing null value with the string value          // throw exception and Print          System.out.println(a.equals("GFG"));      }  } |

1. Discuss the differences between checked and unchecked exceptions. ( CO3)

**Answer :-**

|  |  |  |
| --- | --- | --- |
| Checked Exception | Unchecked Exception | |
| Checked exceptions occur at compile time.  These are also compile time exceptions | | Unchecked exceptions occur at runtime.  These are also called as runtime exceptions |
| The compiler checks a checked exception. | The compiler does not check these types of exceptions. | |
| These types of exceptions should be handled at the time of compilation. | These types of exceptions cannot be a catch or handled at the time of compilation, because during run time the exception is being caught. | |
| They are the sub-class of the exception class. | They are runtime exceptions and hence are not a part of the Exception class. | |
| Here, the JVM needs the exception to catch and handle. | Here, the JVM does not require the exception to catch and handle. | |
| Examples of Checked exceptions:   * File Not Found Exception * I.O Exception * Remote Exceptions * Interrupted Exception | Examples of Unchecked Exceptions:   * Arithmetic Exception * Null Pointer Exception * Array Index Out of Bounds Exception * Class cascade exception * Illegal argument exception | |

1. Why do you use throws clause in exception handling? Explain with an example. ( CO3)

1)if u there are chance of might raise an exception one uses throw(ex see in case of using sleep method on Thread there might occurs an interruption so in that case the method which has Thread.sleep is decaleared as method throws Interrupted exception) us

2)usually w.r.t checked exceptions throws is used

3)u can declare like a method might throw multiple errors using the throws while the exception are separeation by comma(,)

Ex

Psvm() throws InterruptedException,RemoteException,IOExceptions

//thread.sleep might through an interrupted exception

public class hadlingcheckedexception

{

    public static void main(String[] args) throws InterruptedException

    {

        Thread.sleep(10000);

    }

}

1. Write short notes on:1.finally block 2.throws 3.throw



**Answer :- Finally block :-**

**Usually finally block is declared after try and catch .u can use it directly after try also DIRECTLY without catch.working scenario of finally is like the statements in the block will be excecuted even if exception arises and also even if exception not arises.**

The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.

 Using a finally block allows you to run any clean up-type statements that you want to execute, no matter what happens in the protected code.

Say for example if u are working something w.r.t servere u want break connection even after completion of process and also in the case of exception then in those kind of scenarios one uses finally

try {

} catch (ExceptionType1 e1) { // Catch block

} catch (ExceptionType2 e2) { // Catch block

} catch (ExceptionType3 e3) { // Catch block

}finally {

// The finally block always executes.

}

**2. Throws :-**

**See when comes to throws throws is generally used when we know that an exception might arise in a method and u don’t need to do something with respect to that exception then we uses the throws**

If a method is capable of causing an exception that it does not handle, it must specify this behavior so that callers of the method can guard themselves against that exception. We do this by including a **throws** clause in the method’s declaration. A **throws** clause lists the types of exceptions that a method might throw.

type method-name(parameter-list) throws exception-list {

// body of method }

**3. Throw :-**

**See when it comes to throw if u wanted to raise a particular exception with the mssg we uses throw**

The throw keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either checked or unchecked exception or even userdefined exception. The throw keyword is mainly used to throw custom exceptions.

Ex;

Throw new ArithmeticException(“hasi”);

|  |
| --- |
| public class throw1 {      public static void main(String[] args) throws ArithmeticException      {          throw new ArithmeticException("srry");      }  } |

Output:

Exception in thread "main" java.lang.ArithmeticException: srry

at throw1.main(throw1.java:4)

1. Explain the differences between throw and throws keywords in java ( CO3)

**Answer :-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. no.** | **Basis of Differences** | **throw** | **throws** |
| 1. | Definition | Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code. | Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code. |
| 2. | Syntax | The throw keyword is followed by an instance of Exception to be thrown.  s | The throws keyword is followed by class names of Exceptions to be thrown. |
| 3. | Declaration | throw is used within the method. | throws is used with the method signature. |
| 4. | Internal implementation | We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions. | We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException. |
| 5 | When we use | When we explicitly wanted to explicitly rise an exceptions | We wanted not to do anything w.r.t exception we use throws and we we are in a scenario of it might throw an exception and not wanted to do anything explicitlys |

1. Explain with asn example, how to create and handle user defined exception. ( CO3)

|  |
| --- |
| class hi extends Exception  {      hi(String s)      {          super(s);      }  }  public class userdefinedException  {      public static void main(String[] args) throws hi      {          throw new hi("hansi");      }  }  Output:  Exception in thread "main" hi: hansi  at userdefinedException.main(userdefinedException.java:13) |

Another ex:using try catch to handle userdefined exception

|  |
| --- |
| class hiiii extends Exception  {      hiiii(String s)      {          super(s);      }  }  public class userdefinedex  {      public static void main(String[] args)      {          try          {              throw new hiiii("hi exception");          }          catch(hiiii e)          {              System.out.println(e.getMessage());          }        }  }  s |

Output

hi exception