**EXCEPTIONS**

Exception is an exceptional event that occurs during execution of program, that disturbs the normal flow of program.

Exception is an abnormal behavior of an application, occurs when the execution time and terminates the flow of execution if not handled. Exceptions may be database connection errors, network connection errors, file not found error, file corrupted error and so many.

1. What is an exception object and throwing an exception?

When an error occurs within the method, the method creates an object and handles it to the run time to handle, the object is called exception object. Which contains information about the exception.

The object is created and handles (delegates) it to the run time system to handle is called throwing an exception.

After the method throws an exception the run time system looks for the caller of the method to handle, those methods are called call stack.

2. How to handle exceptions?

Exceptions handled by try, catch, and finally blocks.

***Try:*** Try block is used to enclose the code that might throw an exception.

***Catch***: The block catches all the exceptions that have occurred in the try block. Each catch block is an exception handler that handles the exceptions. It can contain specific type of exception to handle that occur in the try block or subclasses of the exception type.

***Finally****:* this block always performs irrespective of exception is catching or not.

3. Difference between error and exception?

Errors typically happen when an application is running while the exception is occurred by the application.

|  |  |
| --- | --- |
| ***Exception*** | ***Error*** |
| Belongs to java.lang package | Belongs to java.lang |
| Exceptions are created by application itself. | Errors are occurred when application is running. |
| Compiler will check for checked exceptions whereas it will not care about the un-checked exceptions. | Compiler will not check errors because they are occurring at run-time. |

4. What is the use of exception handling?

If there is no exception handling, then the application execution terminates. To avoid this, exception handling is necessary.

* Avoids the unexpected termination of program.
* Provides flexibility to the programmers to handle exceptions thereby making application more robust and immune to unexpected errors.
* Programmers can understand the reason for the termination of applications.

5. Types of exceptions in java?

There are two types of exception are present checked and un-checked exceptions.

The parent most of the exception is throwable.

Exception

Error

throwable

***1)Checked exceptions:***

Occurs during compilation time, compiler will check whether the exception is handled, if not then throws an error accordingly.

***2)Unchecked exceptions:***

Occurs during application execution time. Compiler will not check whether the exception is handled or not. It is compiled and throws exception at the run time. All the subclasses of java.lang.RuntimeException class and java.lang.Error classes belong to runtime exceptions.

6. Briefly explain the usage of throw and throws keys?

Both throw and throws are the keys of exception handling.

***throw:*** The throw key is used to explicitly throw an exception inside the method or block of code. The exception is handed over to the run time to handle it.

***throws:*** The throws key is used to delegate or declare which exceptions are thrown by the method. Is used with method signature specifies that method may throw an exception.

7. What is the difference between final, finally and finalize keywords?

*FINAL:*

* Final is a keyword used with variables, methods, and classes, when variable is marked as final it is like a constant it cannot modify in future.
* If the method is final, then it cannot be inherited by the subclass means final methods are not overridden.
* If the class is final then it does not allow inheritance, classes cannot inherit the final classes.

*FINALLY:*

* The finally is a keyword used in the exceptions.
* It executes the statements irrespective of exception is handled or not.

*FINALIZE:*

* The finalize method is called by the java garbage collector when the object is no longer needed.

8. What are the kinds to handle exceptions?

There are multiple kinds of catching the exceptions.

* try with catch
* try only
* try with finally
* try, catch, and finally
* try with multiple catch
* try with multi catch
* try with resource

9. How to handle checked exceptions?

Checked exceptions are also called compile time exceptions; compiler will check whether the exception is handled or not. If not handled properly the program will not be compiled. Can be handled by try-catch block or throws clause in the method declaration.

10. Differentiate between checked and unchecked exception?

|  |  |
| --- | --- |
| ***Checked exceptions*** | ***Unchecked exceptions*** |
| They are checked by the compiler. | They are not checked by the compile. |
| Until and unless exception is handling the compilation of the program is not possible. | Compilation may possible but exception may arise during runtime if exception is not handled properly. |
| Checked exceptions are subclasses of Exception class. | They are subclasses of RuntimeException class. |
| Ex: IOExceptions, FileNotFoundException etc.. | Ex: ArithmeticException, NullPointerException etc.. |

11. What is stack trace and how it is related to exception?

Stack trace is information consisting of names of classes and methods that were invoked right from the start of the program execution to the point where an exception occurred. This is helpful to know where the exception occurred and reason for exception.

12. What are the methods of the Exception class?

Exception class several methods below they are:

*getMessage()* :- returns a string that describes the exception.

*printStackTrace()* :- prints stack trace of the exception to the console.

*equals()* :- compares two exceptions for equality.

*toString()* :- returns string representation of the exception.

*clone()* :- creates clone or copy the exception.

*hashCode()* :- returns hash code for the exception.

*getCause()* :- returns cause of the exception if available.

*getStackTrace()* :- returns an array of StackTarceElement objects that represents the stack trace of the exception.

*initCause()* :- initializes the cause of the exception.

13. What is exception chaining?

One exception is thrown due to another exception called exception chaining. This helps developers to identify what situations an exception was thrown that in turn cause another exception.

14. Explain unreachable catch block?

This error is thrown by the compiler when multiple catch blocks and keep parent classes first and subclasses later. The catch block should follow the order, the specific ones (subclasses) are at the top and general ones (parent classes) are at the bottom. If not followed, an unreachable catch block error is thrown during compile time.

15. Explain try with resource block?

The try-with-resource statement is used for resource management, the resource that is opened in the try block will be automatically closed when try block finishes.

In earlier versions resources are usually closed in the finally block.

try (resource initialization) {

}

catch () {

} // finally block in not needed.

The resource that is declared inside the parenthesis of the try block must implement the AutoCloseable interface or Closeable interface.

16. NullPointerException?

* It is a *Runtime Exception* (extends Runtime Exception).
* A special null value can be assigned to an object reference.
* When using the reference of that object throw a ***NullPointerException***.
* These include.
  + Calling the instance method of a null object.
  + Accessing or modifying the field (variable) of the null object.
  + Taking length of null as if it were an array.
  + Accessing or modifying the slots of null as if it were an array.
  + Throwing null as if it were throwable value.

17. How to resolve NullPointerException?

To avoid *NullPointerException*, we must ensure that all *objects are initialized* *properly* before their usage. When declaring a reference variable, we must ensure that object is not null before we access a field or method of the object.

18. Explain *Errors* in java?

Error is an ***illegal operation*** performed by the developer which leads to ***abnormal*** working of program. Errors remains undetected until the program is compiled or executed. Some of the errors will restrict the program getting compiled until they are not removed.

19. What are the types of errors?

There are 3 types of errors are there:

1. *Run-time Errors*
2. *Compile time Error*
3. *Logical Errors*

20. Explain Run-time Errors?

Runtime errors occur during the execution of the program. Sometimes these are discovered when the user enters the invalid data.

21. Explain *CloneNotSupportedException*?

If the object on which clone() method invoke implement the Cloneable interface. If the object does not cloned, then is arises the exception called *CloneNotSupportedException*.

**SERVLETS**

1. What is servlet?

It is a class used to extend capabilities of server that host applications by means of request response programing model. Servlets dynamically extend the functionality of the web server.

2. What is servlet? Explain in brief.

* Servlet is an interface.
* Servlet is a small java program which runs within web server.
* Servlet receives and responds to requests from the clients across http.
* Which comes under javax package.
* There are two ways to implement this servlet one is by implementing Servlet interface with GenericServlet class and another way is by extending GenericServlet to HttpServlet class.
* This Servlet interface defines methods to initialize servlet, to service requests and to remove a servlet from the server.
* These methods are called life cycle methods of servlet.

3. Life-cycle methods of servlet?

* The servlet is constructed then initialized with *init()* method.
* Any calls from clients to the *service()* method are handled.
* If the servlet is taken out of service, then destroyed with the *destroy()* method and then garbage is collected and finalized.

In addition to life-cycle methods servlet also provides *getServletConfig()* by this servlet can be used to get any startup information, *getServletInfo()* gives the basic information about the servlet itself such version, author.

3. Explain Servlet API?

A Servlet does not have main method instead of that Servlet has methods for purpose of handling requests. It invokes service methods every time Server sends a request to Servlet. To handle requests Servlet must override service method that allow two parameters these are request object and response object. Request object is used to inform the servlet about the request and response object is used to give the response.

4. Define init and destroy methods of servlet?

Servlet Init method is used to initialize the servlet.

After the web container loads and instantiate the Servlet class and before it delivers requests from clients, the web container initializes the servlet. This process allows the servlet to read configuration data, initialize process, and perform any other one-time activities.

When a servlet container determines that servlet should be removed from the service, the container calls the destroy method to destroy the service.

5. What are the methods of Servlet?

A screenshot of a computer

Description automatically generated

**WRAPPER CLASSES**

1. What is wrapper class?

Wrapper classes are classes that allow us to access primitive datatypes as objects.

Ex: Integer is wrapper class it is used to represent primitive int type. We can use wrapper classes to represent primitive types.

Wrapper classes are Byte, Short, Integer, Float, Double, Long, BigInteger, BigDecimal.

2. What are the uses of wrapper classes?

* Wrapper classes allow us to create an object.
* We can add them to collections and pass them as arguments to methods.
* They provide methods to manipulate data and the ability to convert to different types.
* They help to improve performance.
* Ex: when we use autoboxed Integer object, the java compiler can convert the object to primitive int value automatically.
* Because compiler does not have to generate code to convert object to primitive value.

Integer myInt = new Integer(10);

int myInt2 = myInt.intValue(); // Converts the Integer object to a primitive int value

3. How to convert String type to primitive datatype?

It is possible to convert String datatype to primitive types using static *parse()* methods provided by the wrapper classes.

4. What is difference between *parseInt()* and *valueOf()* methods?

The parseInt() is used to convert String datatype to primitive type, whereas valueOf() is used to convert object into a String.

5. Are all wrapper classes immutable?

No, not all wrapper classes are immutable. Only Byte, Boolean, Short, Character are immutable. Integer, Float, Double, Long are mutable.

6. What does the compareTo() method do?

The *compareTo()* method is used to compare two wrapped objects. It returns 0 if two objects are same, positive 1 if the first object is greater than second object and negative -1 if the first object is less than second object.

7. Explain autoboxing?

Autoboxing in java is the automatic conversion of primitive type into their corresponding wrapper classes. With autoboxing the compiler automatically converts primitive types into wrapper classes.

Ex: int a = 10;

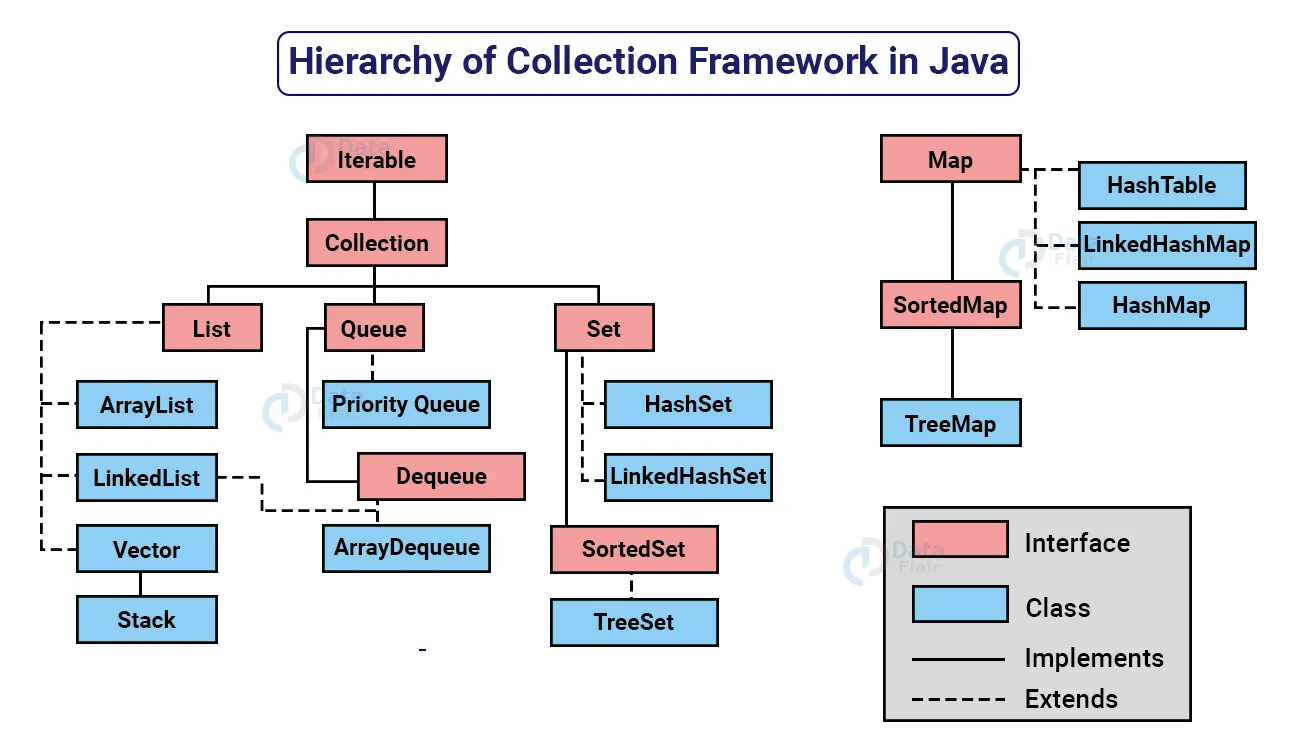
Integer b = a; //autoboxing

**COLLECTIONS**

Collection is like a container, an object that groups multiple elements into to a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data. They represent data items that form natural group.

Ex: Telephone directory, Mail folder, Poker cards.

Collection hierarchy:



1. What is the collections framework?

Is a unified architecture for representing and manipulating collections. Collection framework contains interfaces, implementations, and algorithms.

* *Interfaces:* Allow collections to be manipulated independently of the details of their representations.
* *Implementation:* These are the concreate implementations of the collection interface, they are reusable data structure.
* *Algorithms:* These are methods to perform operations such sorting and searching.

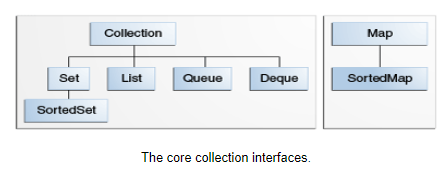
2. What are the benefits of collections framework?

1. ***Reduces programming efforts:*** provides useful data structures and algorithms to manipulate collections.
2. ***Increases programming speed and quality:*** collections framework provides high-performance, high-quality data structures and algorithms to manipulate.
3. ***Allows interoperability among unrelated API’s:*** collections frameworks provide interoperability among unrelated APIs to perform operations.
4. ***Reduces effort to learn and use new API’s:*** earlier there are lot of APIs are present to work with collections but know by using efficient APIs this problem resolved.
5. ***Reduces effort to design new API’s:*** no need to design new APIs to work with collections instead of using standard collections interfaces.
6. ***Foster software reuse:***

3. Explain Collections class?

*java.util.Collections* is a class consisting of static methods (algorithms) to perform on the collections. This class contains methods for algorithms, like binary sorting, search, shuffling etc.

4. Explain Collection interface?



Collection interface is the root hierarchy of the java collection framework. It encapsulates different types of interfaces that are independent of their representation. Collection interface provides methods to perform operations on the collections.

5. Explain Iterator interface?

Iterator is an interface that contains two abstract methods and one non-return (void) method like hasNext(), next() and remove() respectively. This interface allows collections to iterate over the collection. There are three ways to iterate a collection they are Aggregate operation, for each Construct, and iterator.

Iterator provides the iterator method to iterate the collection and provides the remove() method to remove the element that satisfies the condition during iteration.

6. Explain Set interface?

A Set is a collection interface that restricts the duplicate elements stored in the collection. It does not maintain insertion order of the elements. It contains methods that are inherited from the Collection interface and uses those methods to restricts duplicate elements by using hashCode() and equals() methods.

Set interface has three implementations:

* HashSet: It stores elements in hash table, it makes no guarantees concerning order of the iteration.
* TreeSet: It stores elements in red-black tree, orders based on the values of the elements.
* LinkedHashSet: Which is implemented as a hash table with linked list running through it. It stores elements based on the insertion order.

7. What is ArrayList?

ArrayList is a data structure used to elements that have dynamic behavior i.e., it can grow its size by adding elements and can shrink size by removing elements.

8. What are the methods of Collection?



A screenshot of a computer

Description automatically generated

9. Explain List interface?

List is the data structure of the collection framework. List stores multiple items based on insertion order; it extends methods from the collection to operate on the collections. It has two implementations they are ArrayList and LinkedList. It can contain duplicate items also.

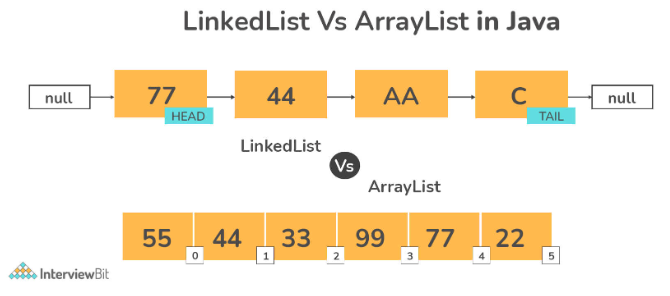
10. Explain Queue interface?

Queue is a data structure, besides basic collection operations queue provides additional insertion, removal, and inspection operations. Queue typically, but not necessarily orders elements in FIRST IN FIRST OUT (FIFO) manner.

11. Explain Map?

Map is an object that maps keys to values, it can’t contain duplicate keys, one has map to at most one value. It has methods (put, get, remove, containsKey, containsValue, size, and empty) to perform basic operations on map data, for the bulk operations we use clear, putAll methods, for collection view keyset, entrySet, and values are used.

12. List out differences between ArrayList and LinkedList?



|  |  |
| --- | --- |
| ***ArrayList*** | ***LinkedList*** |
| Elements of this class are stored in a dynamic array. | Elements of this class are stored in a doubly linked list. |
| This class implements List interface. As a result, it can be used as List. | This class implements both List and Deque as a result it can be used as both List and Deque. |
| Because of internal implementations, manipulating ArrayList takes longer time. Internally ArrayList is scanned, and memory are shifted whenever we remove an element. | There is no internal implementations on LinkedList, so it doesn’t take much time to manipulate. The reference link is changed after traversing the list. |
| This class is more useful when the application requires the data storage and access. | This class is more useful when the application requires data manipulation. |

13. Differentiate between ArratList and Vector?

|  |  |
| --- | --- |
| ***ArrayList*** | ***Vector*** |
| Can’t be synchronized | Can be synchronized |
| It is not a legacy class | It is a legacy class |
| Not thread safe | Thread safe |

14. Difference between Iterator and ListIterator?

|  |  |
| --- | --- |
| ***Iterator*** | ***ListIterator*** |
| Used to traverse array items in forward direction only | Used to traverse array items in both directions. |
| Can be used with Queue, List, Set | Used with List |
| Performs only remove operation | Can perform operations like add, remove, set operations while traversing the collection |

15. Difference between List and Set?

|  |  |
| --- | --- |
| ***List*** | ***Set*** |
| List maintains insertion order. | Set doesn’t maintain insertion order. |
| Duplicate elements are allowed in list. | Duplicate elements are not allowed. |
| List is an index based. | Set is not an index based. |
| Elements can accessed by their index. | Set not allow access to their elements from their certain position. |
| It is possible to store null elements several times. | A null value can stored only once. |

16. Differentiate between HashSet and TreeSet?

|  |  |
| --- | --- |
| ***HashSet*** | ***TreeSet*** |
| HashSet doesn’t provide any ordering guarantee. | TreeSet provides ordering or sorting guarantee. |
| HashSet uses equals() method for comparison and avoids duplicates. | TreeSet uses compareTo() method for comparison. |
| HashSet uses HashTable as data structure. | TreeSet uses Red-Black Tree. |
| HashSet allows one null element. | TreeSet doesn’t allow null objects. |
| Implemented using HasMap. | Implemented using TreeMap. |
| HashSet is faster. | TreeSet is slower. |

17. Can we add null element to the TreeSet?

No. We can’t add null element to the TreeSet because it uses compareTo() method for comparing, it throws NullPointerException if it contains null element.

**JAVA DATABASE CONNECTIVITY (JDBC)**

JDBC is a Java API used to make things easy. Which is used to execute common SQL queries and operate on the database.

JDBC API is Java API that can access any kind of tabular data that is stored in relational database. It uses JDBC Drivers to connect with database.

**JDBC helps to write Java applications that manages activities:**

1. *To connect to database.*
2. *Send queries and update statements to the database.*
3. *Retrieve and process the data received from the database.*

**Components of JDBC:**

*JDBC API****:*** The JDBC API provides methods and interfaces to access relational data using java programming language. Using API also allowed to execute SQL queries, retrieve data, and process the data.

*JDBC DriverManager****:*** The JDBC DriverManager class defines objects which can connect to JDBC driver. It loads JDBC Driver in java applications for establishing connection with the database.

*JDBC Test Suite:* It is used to test the operations like updating, deleting, inserting etc., being performed by the JDBC Drivers.

*JDBC-ODBC Bridge Drivers:* It will connect database drivers to the database. JDBC-ODBC bridge interprets JDBC method call to the ODBC function call.

***JDBC ARCHITECTURE:***

JDBC API supports both *two tier* and *three tier* processing model for database access.

A diagram of a software application

Description automatically generated

*1) Two tier Architecture:*

* In Two tier model the java applications directly communicate with data source (database).
* JDBC drivers helps applications to communicate with the data source.
* A user’s commands are delivered to the data source (database) and results are sent back to the user.

A diagram of a data access

Description automatically generated

*2) Three tier Architecture:*

A diagram of a computer server

Description automatically generated

* In three tier architecture model the ***middle tier*** exists in between the user and the database.
* The middle tier acts as a service provider which receives the commands from the user and sends those to the database and vice versa.
* Three tier model is more effective than two tier because it has a middle tier.
* Advantage is it maintains control over the data access.

***STEPS TO PROCESS SQL STATEMENT WITH JDBC:***

1. *Establish connection.*
2. *Create a statement.*
3. *Execute the query.*
4. *Process the result set object.*
5. *Close the connection.*

***Establishing connection:*** First, need to establish connection with the data source. Data source may be any database, which needs to be use or any external file that corresponds to Driver.

* DriverManager: It is an implementation class; it connects an application with the data source (database) which is specified by the database URL.
* Connection is done using method DriverManager.getConnection().

***Create statement:*** To create statement object, connection object is needed. Statement is an interface that represents SQL statements.

There are 3 different kinds of statements.

1. *Statement:* statement used to implement simple SQL statements.
2. *PreparedStatement:* extends Statement used for precompiling SQL statements that might contain input parameters.
3. *CallableStatement:* extends PreparedStatement used to execute stored procedure that may contain both input output parameters.

***Execute Queries:*** Queries can be executed by calling execute() method of the Statement such as the following

* *execute()* : Returns true if the first object that the query returns ResultsSet object. Return type is Boolean.
* *executeQuery()* : Returns one ResultsSet object. Return type is ResultSet.
* *executeUpdate()* : Returns an integer representing the number of rows affected by the SQL statement. Return type is int.

*ResultSet:* The ResultSet interface represents the database result set which is obtained after the execution of SQL query using Statement object.

* The object of ResultSet maintains cursor position to the current position of row of the data.
* Initially the cursor is located before the first row. Then the cursor is moved to next row using next() method.
* To iterate over the rows of the data next() method is used.

1. What is stored Procedure? What are the parameters types in stored Procedure?

Stored procedures are group of SQL queries that are executed as single logical unit to perform specific task. Name of the procedure should be unique since each procedure is represented by its name.

Three types of parameters types are presented in stored procedure.

* IN: It is used for passing input value to procedure with help of setXXX() method.
* OUT: It is used to get value from the procedure using getXXX() method.
* IN/OUT: it is used for passing the input value and obtaining the value to/from the procedure.

2. What is DatabaseMetaData?

* DatabaseMetaData is an interface that provides methods to obtain information about the database.
* Information like database name, database version, driver name, total number of tables or views, et.

3. List out the differences between JDBC and ODBC?

|  |  |
| --- | --- |
| ***ODBC(OPEN DATABASE CONNECTIVITY)*** | ***JDBC(JAVA DATABASE CONNECTIVITY)*** |
| Used for programing languages like C, C++, Java, etc. | Used only for Java language. |
| ODBC is platform dependent. | JDBC is platform independent. |
| Most of the ODBC drivers are developed in native languages like C, C++ | JDBC drivers are developed using Java language. |
| Not recommended to use ODBC for java application because of low performance. | JDBC is recommended for Java application. |
| ODBC is procedural. | JDBC is object oriented. |