**JAVA**

# String

Immutable - unmodifiable or unchangeable

## String Compare

Three ways to compare string in java:

1. By equals() method
2. By = = operator
3. By compareTo() method

### equals()

* **public boolean equals(Object another)** compares this string to the specified object.
* **public boolean equalsIgnoreCase(String another)** compares this String to another string, ignoring case.

## StringBuffer class

used to created **mutable** (modifiable) string

StringBuffer class is thread-safe i.e. multiple threads cannot access it simultaneously

## Difference between StringBuffer and StringBuilder

|  |  |  |  |
| --- | --- | --- | --- |
|  | String | StringBuffer | StringBuilder |
| Modifiable | No (immutable) | Yes( mutable ) | Yes( mutable ) |
| Thread Safe | Yes | Yes | No |
| Performance | Fast | Very slow | Fast |
| Storage Area | Constant String Pool | Heap | Heap |
|  |  | Synchronized | Non Synchronized |

## Create Immutable class



 Class is immutable because:

* The instance variable of the class is final i.e. we cannot change the value of it after creating an object.
* The class is final so we cannot create the subclass.
* There is no setter methods i.e. we have no option to change the value of the instance variable.

# Super Keyword

Reference variable that is used to refer immediate parent class object.

1. super is used to refer immediate parent class instance variable.
2. super() is used to invoke immediate parent class constructor.
3. super is used to invoke immediate parent class method.



# Final Keyword

* Compile time error message when we try to update final variable
* Compile time error - Final method cannot be override



* class as final, cannot be extended.
* A final variable that is not initialized at the time of declaration is known as blank final variable.



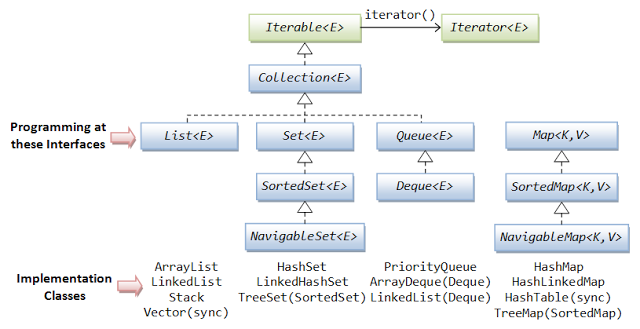
* Final as parameter – we cannot change final value



* Constructor cannot be declared as final

# ABSTRACTION

# Collection



### Why does Map not extend Collection interface

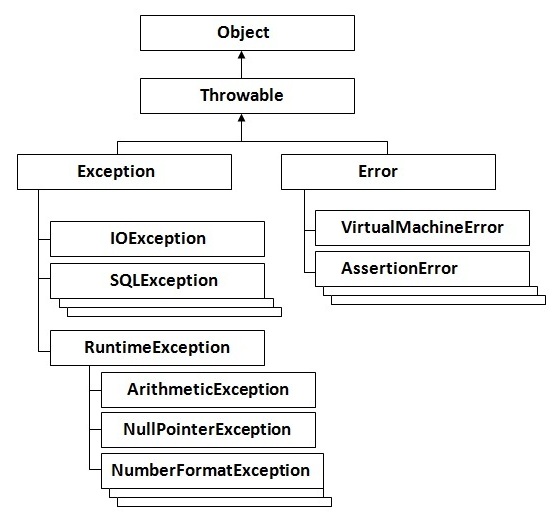
Reason for Map interface not extending Collection interface

* If you look at the respective data structure you can easily guess why Map is not a part of Collection. Each Collection stores a single value where as a Map stores key-value pair. So methods in Collection interface are incompatible for Map interface.For example in Collection we have add(Object o).What would be such implementation in Map. It doesn't make sense to have such a method in Map. Instead we have a put(key,value)  method in Map.
* Same argument goes for addAll(), remove(), removeAll() methods. So the main reason is the difference in the way data is stored in Map and Collections.
* Also if you recall Collection interface implemented Iterable interface i.e any interface with .iterator() method should return an iterator which must allow us to iterate over the values stored in the Collection. Now what would such method return for a Map? Key iterator or a Value iterator? This does not make sense either.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Duplicate** | **Element Ordered** |  |
| List |  |  |  |
| Set |  |  |  |
| Queue |  |  |  |
| Map |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Collection | Ordering | Random Access | Key-Value | Duplicate Elements | Null Element | Thread Safety |
| ArrayList | Yes | Yes | No | Yes | Yes | No |
| LinkedList | Yes | No | No | Yes | Yes | No |
| HashSet | No | No | No | No | Yes | No |
| TreeSet | Yes | No | No | No | No | No |
| HashMap | No | Yes | Yes | No | Yes | No |
| TreeMap | Yes | Yes | Yes | No | No | No |
| Vector | Yes | Yes | No | Yes | Yes | Yes |
| Hashtable | No | Yes | Yes | No | No | Yes |
| Properties | No | Yes | Yes | No | No | Yes |
| Stack | Yes | No | No | Yes | Yes | Yes |
| CopyOnWriteArrayList | Yes | Yes | No | Yes | Yes | Yes |
| ConcurrentHashMap | No | Yes | Yes | No | No | Yes |
| CopyOnWriteArraySet | No | No | No | No | Yes | Yes |

# Exception Handling



### **Types of Exception**

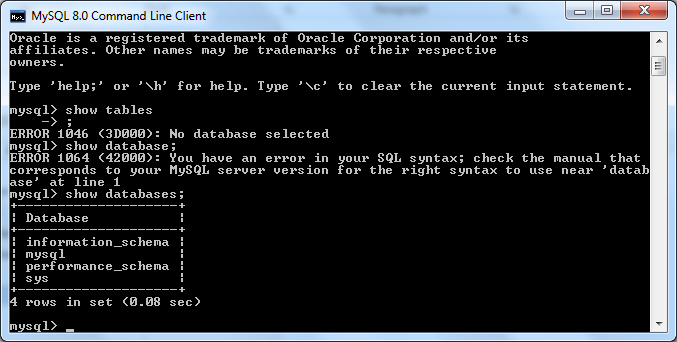
1. Checked Exception
2. Unchecked Exception
3. Error

#### Note: The finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort).

# MySql

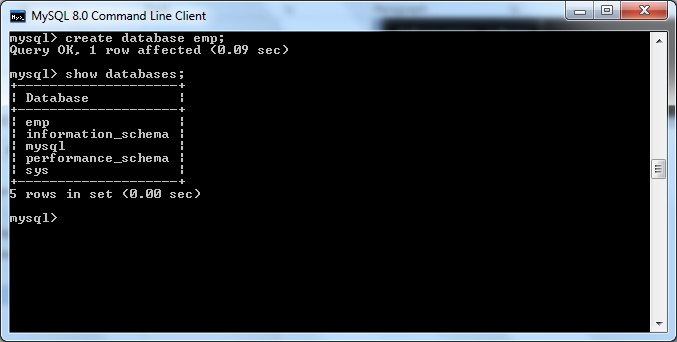
## Show database

Mysql> show databases;



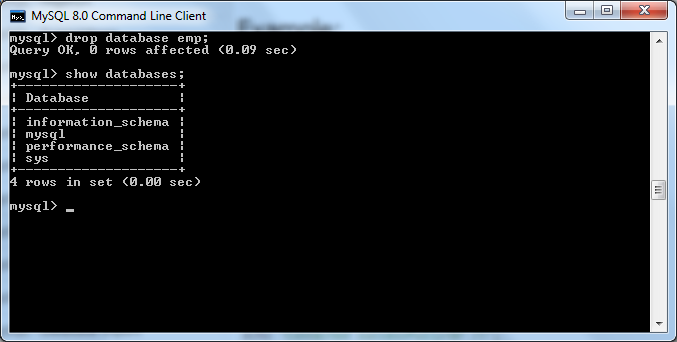
## Create database

Mysql> create database <Database Name>;

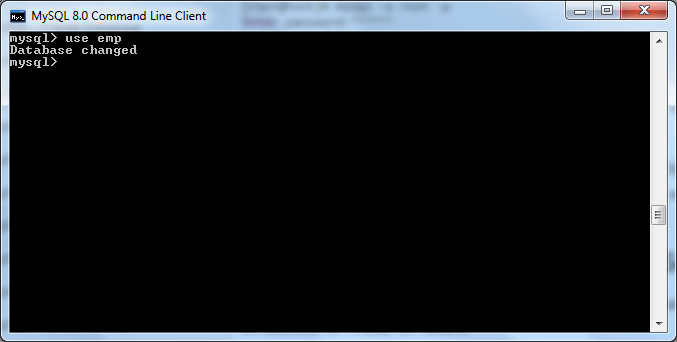


## Drop database

Mysql> drop database <Database Name>;



## Select database

Mysql>use <Database Name> 

## Create table

CREATE TABLE table\_name (column\_name column\_type);

# SQL

## **INTERVIEW**

1. Define join and name different type of joins?
2. What is the syntax to add record to a table?
3. How do you add a column to a table?
4. [Define Delete, drop, truncate?](#_Delete,_Drop,_Truncate)
5. Definitions of DBMS Keys
6. How to select random rows from a table?
7. Write a SQL SELECT query that only returns each name only once from a table?
8. Suppose a Student column has two columns, Name and Marks. How to get name and marks of top three students
9. What is difference between Having clause and Where clause?
10. Find 3rd Max/Min salary of employee.
11. How do you change value of the field?
12. About View, Index, triger

## **DDL, DML, DCL, TCL**

|  |  |  |  |
| --- | --- | --- | --- |
| DDL | DML | DCL | TCL |
| Data Definition Language | **Data Manipulation Language** | **Data Control Language** | **Transaction Control** |
| CREATE - to create objects in the database  ALTER - alters the structure of the database  DROP - delete objects from the database  TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed  COMMENT - add comments to the data dictionary  RENAME - rename an object | SELECT - retrieve data from the a database  INSERT - insert data into a table  UPDATE - updates existing data within a table  DELETE - deletes all records from a table, the space for the records remain  MERGE - UPSERT operation (insert or update)  CALL - call a PL/SQL or Java subprogram  EXPLAIN PLAN - explain access path to data  LOCK TABLE - control concurrency | GRANT - gives user's access privileges to database  REVOKE - withdraw access privileges given with the GRANT command | COMMIT - save work done  SAVEPOINT - identify a point in a transaction to which you can later roll back  ROLLBACK - restore database to original since the last COMMIT  SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use |
|  | Manage data within DB objects | Control the access of data stored in database and provide security. | Manage changes made by DML statements |

## **Delete, Drop, Truncate**

|  |  |  |
| --- | --- | --- |
| DELETE | TRUNCATE | DROP |
| command is used to remove rows from a table | removes **all rows** from a table | removes a table from the database |
| A WHERE clause can be used to only remove some rows. If no WHERE condition is specified, all rows will be removed. | cannot be rolled back and no triggers will be fired | The operation cannot be rolled back. |
| After performing a DELETE operation you need to COMMIT or ROLLBACK the transaction to make the change permanent or to undo it | TRUCATE is faster and doesn't use as much undo space as a DELETE. | No DML triggers will be fired. |
|  | SQL> TRUNCATE TABLE emp;  Table truncated | SQL> DROP TABLE emp;  Table dropped. |
| DELETE is a DML command | TRUNCATE is DDL command | DROP is DDL command |
| DELETE operations can be rolled back | TRUNCATE operations cannot be rolled back. | DROP operations cannot be rolled back. |
|  |  |  |

## **SQL JOIN**

INNER JOIN is the same as JOIN; the keyword INNER is optional



# SELENIUM

## **INTERVIEW**

1. Why should Selenium be selected as a test tool?
2. What are the limitations of Selenium?
3. What are the different types of locators in Selenium?
4. What difference is between assert and verify commands?
5. What is an Xpath?
6. What is the difference between “/” and “//” in Xpath?
7. How do I launch the browser using WebDriver?
8. What are the different types of waits available in WebDriver?
9. About Selenium Commands?
10. How can we get a text of a web element?
11. How to select value in a dropdown?
12. What are the different types of navigation commands?
13. How to handle frame in WebDriver?
14. When do we use findElement() and findElements()?
15. How to find more than one web element in the list?
16. What is the difference between driver.close() and driver.quit command?
17. How to Handle Alert, Popup, Iframe
18. How to mouse hover on a web element using WebDriver?
19. How to capture screenshot in WebDriver?
20. What are Junit annotations?
21. What are the TestNG annotations?
22. How to set test case priority in TestNG?
23. How can I read test data from excels?
24. Explain POM?
25. What is the difference between Selenium and QTP?
26. What is Object Repository? How can we create Object Repository in Selenium?
27. How to check element is visible or not / enabled or not

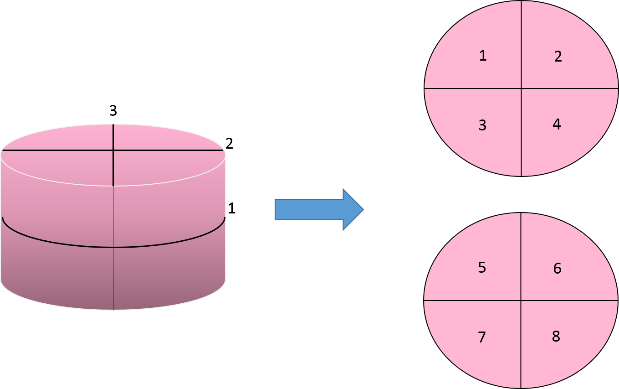
# Puzzle

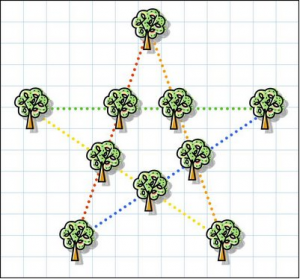
1. A Milkman has 2 Empty Jugs and milk size of Jug A is 3 ltr and Jug B is 5 Ltr how can he measure exactly one liter without wasting any milk.
2. A man needs to cross a river in a canoe, with him, he has a bag of grain, a chicken and a fox. He can only carry one of three at a time. If he leaves the grain and the chicken, the chicken will eat the grain. If he takes the grain, the fox will eat the chicken. How does he successfully cross the river with his load?
3. You have a birthday cake and have exactly 3 cuts to cut it into 8 equal pieces. How do you do it?
4. How do you plant 10 trees in 5 rows with 4 trees in each row?
5. You have 8 coins, all are identical in their shape and size, but one of them is lighter. How u can find the lighter one but the worst case should not be more than 2.
6. There is a pond, and you have 2 container of 5 and 3 liters. You need to provide me 4 liters of water. How would you do that?
7. There are 25 horses. We have to find out the fastest 3 horses. In one race maximum 5 horses can run. How many such races are required in minimum to get the result?
8. There is a boat, which is completely filled with fuel such that even if a bird comes and sit over then it cannot bear its weight and will get sink. Now after running few kilometers an eagle came and sat over it but the boat did not sunk.
9. A black man dressed all in black, wearing a black mask, stands at a crossroads in a totally black-painted town. All of the streetlights in town are broken. There is no moon. A black-painted car without headlights drives straight toward him, but turns in time and doesn’t hit him. How did the driver know to swerve?
10. There are three boxes. One is labeled apples, another is labeled oranges. The last one is labeled apples and oranges. You know that each one is labeled incorrectly. You may ask me to pick one fruit from one box which you choose. How can you label the box correctly?
11. You have a 9 ball,8 ball's weight is same and one is different. how to find that ball in minimum step.

ANSWER

1. The milkman filled the three gallon jug, and then emptied the contents into the five gallon jug. He then filled the three gallon jug again, and continued to fill the five gallon jug until it was full. The milk remaining in the three gallon jug was precisely one gallon.

3.





1. Make a group of 3, 3 and 2 balls. Keep that set of 3 and 3 on pans of weighing machine.   
   **Case1:** if the weigh is same. Good enough, you are left with 2 balls and in the next weighing you weigh them to get your answer.  
   **Case2:**if they weigh differently. You need at-least to have an information about the odd ball being lighter or heavier.
2. Fill 5 liter container

Empty 5 liter in 3 Liter container. Empty the 3 liter container.

Empty 5liter container in 3 liter

Fill 5 Liter container and empty the 5 liter container in 3 liter container

|  |  |
| --- | --- |
| 5 Liter container | 3 Liter container |
| 5 | 0 |
| 2 | 3 |
| 2 | 0 |
| 0 | 2 |
| 5 | 2 |
| 4 | 3 |

1. Divide the 9 balls into 3 sets of 3 balls each.   
   **Weighing 1**: set 1 against set 2.  
   **Weighing 2**: set 1 against set 3.

# Manual Testing

1. Explain Type of Testing : Static testing & Dynamic Testing
2. Explain STLC Model
3. Explain Bug Life Cycle
4. Explain Priority & Severity
5. Explain V Model
6. Difference Between: Smoke & Sanity,Verification & Validation ,Retesting & Regression Testing,Alpha testing & Beta Testing,Test Strategy & Test Plan
7. Explain Use Case Testing
8. Explain Test Plan
9. Explain Agile Testing
10. Explain Error guessing & Error Sending
11. Explain Bug Leakage & Bug release
12. Explain BVA & EVP
13. Explain UAT,Test Harness,Test Bad,Statement coverage,Branch Coverage,Exploratory Testing,compatibility testing,
14. About Defect Tracking tools.

# API

## **Response Codes**

|  |  |
| --- | --- |
| 1xx | information based |
| 2xx | Success |
| 3xx | Redirection |
| 4xx | Client Error |
| 5xx | Server Error |

**ancestor**

     Ancestor lets you select any ancestors [e.g., Parent and Grandparent] of the current node.  
     //\*[@id='regular-expression-syntax']/ancestor::div[5]/div[2]  
     //\*[@id='regular-expression-syntax']/ancestor::\*

**ancestor-or-self**

     Ancestor-or-self lets you select any ancestors [e.g., Parent and Grandparent] of the current                      node including the current node.  
     //\*[@id='regular-expression-syntax']/ancestor-or-self::div[1]  
     //\*[@id='regular-expression-syntax']/ancestor-or-self::\*

**attribute**

     Attribute returns all the attributes in the current node.  
     //\*[@class='sphinxsidebarwrapper']/attribute::\*  
     //\*[@id='sidebarbutton']/attribute::title

**child**

     Child returns all the children in the current node.  
     //\*[@class='sphinxsidebarwrapper']/child::\*  
     //\*//child::h3

**descendant**

     Descendant lets you select all descendants [e.g., Children and Grandchildren] of the current node.  
     //\*[@class='this-page-menu']/descendant::\*  
     //\*[@class='this-page-menu']/descendant::li[2]  
     //\*[@class='documentwrapper']/descendant::div[position()=3]

**descendant-or-self**

     Descendant-or-self lets you select all descendants [e.g., Children and Grandchildren] of the current       node including the current node.  
     //\*[@class='this-page-menu']/descendant-or-self::\*  
     //\*[@id='searchbox']/descendant-or-self::form[@class='search']/input[2]

**following**

     Following returns all in the document after the closing tag of the current node.  
     //\*[@class='clearer']/following::\*

**following-sibling**

     Following-sibling returns all the sibling after the closing tag of the current node.  
     //\*[@class='related']/following-sibling::\*  
     //\*[@class='related']/following-sibling::div[3]

**ancestor**

     Ancestor lets you select any ancestors [e.g., Parent and Grandparent] of the current node.  
     //\*[@id='regular-expression-syntax']/ancestor::div[5]/div[2]  
     //\*[@id='regular-expression-syntax']/ancestor::\*

**ancestor-or-self**

     Ancestor-or-self lets you select any ancestors [e.g., Parent and Grandparent] of the current                      node including the current node.  
     //\*[@id='regular-expression-syntax']/ancestor-or-self::div[1]  
     //\*[@id='regular-expression-syntax']/ancestor-or-self::\*

**attribute**

     Attribute returns all the attributes in the current node.  
     //\*[@class='sphinxsidebarwrapper']/attribute::\*  
     //\*[@id='sidebarbutton']/attribute::title

**child**

     Child returns all the children in the current node.  
     //\*[@class='sphinxsidebarwrapper']/child::\*  
     //\*//child::h3

**descendant**

     Descendant lets you select all descendants [e.g., Children and Grandchildren] of the current node.  
     //\*[@class='this-page-menu']/descendant::\*  
     //\*[@class='this-page-menu']/descendant::li[2]  
     //\*[@class='documentwrapper']/descendant::div[position()=3]

**descendant-or-self**

     Descendant-or-self lets you select all descendants [e.g., Children and Grandchildren] of the current       node including the current node.  
     //\*[@class='this-page-menu']/descendant-or-self::\*  
     //\*[@id='searchbox']/descendant-or-self::form[@class='search']/input[2]

**following**

     Following returns all in the document after the closing tag of the current node.  
     //\*[@class='clearer']/following::\*

**following-sibling**

     Following-sibling returns all the sibling after the closing tag of the current node.  
     //\*[@class='related']/following-sibling::\*  
     //\*[@class='related']/following-sibling::div[3]

# Java SE 8 Programmer I 1Z0-808

## Java Basics

* Define the scope of variables
* Define the structure of a Java class
* Create executable Java applications with a main method; run a Java program from the command line; including console output.
* Import other Java packages to make them accessible in your code
* Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.

## Working With Java Data Types

* Declare and initialize variables (including casting of primitive data types)
* Differentiate between object reference variables and primitive variables
* Know how to read or write to object fields
* Explain an Object's Lifecycle (creation, "dereference by reassignment" and garbage collection)
* Develop code that uses wrapper classes such as Boolean, Double, and Integer.

## Using Operators and Decision Constructs

* Use Java operators; including parentheses to override operator precedence
* Test equality between Strings and other objects using == and equals ()
* Create if and if/else and ternary constructs
* Use a switch statement

## Creating and Using Arrays

* Declare, instantiate, initialize and use a one-dimensional array
* Declare, instantiate, initialize and use multi-dimensional array

## Using Loop Construct

* Create and use while loops
* Create and use for loops including the enhanced for loop
* Create and use do/while loops
* Compare loop constructs
* Use break and continue

## Working with Methods and Encapsulation

* Create methods with arguments and return values; including overloaded methods
* Apply the static keyword to methods and fields
* Create and overload constructors; including impact on default constructors
* Apply access modifiers
* Apply encapsulation principles to a class
* Determine the effect upon object references and primitive values when they are passed into methods that change the values

## Working with Inheritance

* Describe inheritance and its benefits
* Develop code that demonstrates the use of polymorphism; including overriding and object type versus reference type
* Determine when casting is necessary
* Use super and this to access objects and constructors
* Use abstract classes and interfaces

## Handling Exceptions

* Differentiate among checked exceptions, unchecked exceptions, and Errors
* Create a try-catch block and determine how exceptions alter normal program flow
* Describe the advantages of Exception handling
* Create and invoke a method that throws an exception
* Recognize common exception classes (such as NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException, ClassCastException)

## Working with Selected classes from the Java API

* Manipulate data using the StringBuilder class and its methods
* Creating and manipulating Strings
* Create and manipulate calendar data using classes from java.time.LocalDateTime,
* -java.time.LocalDate, java.time.LocalTime, java.time.format.DateTimeFormatter, java.time.Period
* Declare and use an ArrayList of a given type
* Write a simple Lambda expression that consumes a Lambda Predicate expression