String:

1. String is not a primitive time. It is a regular java class.
2. String with double quote - string pool .String with new keyword created in heap
3. String objects are equal ?– Hashcode is same.
4. In Java all strings are immutable. Once object is created, it cannot be changed.

All primitive wrapper class are immutable – Boolean,long ,integer etc

Immutable:

Pro: Same object can be used. Memory efficient if same object is used

Thread safe. Many threads can use them

Con:

Small change makes JVM to create new object created. Not memory efficient.

Hence less performance.

How to overcome this?

Java provides String builder and String buffer 🡪manipulate Strings and not create new object

1. Difference between them

Bufffer is threadsafe. Builder – not thread safe. No guarantee of sync.

If same operation is done by String, Stringbuffer&builder - Builder will be fastest.

1. ToString -to print String object with values of its variable [ override the tostring method of Object]

Autoboxing:

1. Wrapper classes are the object representation of primitive types

Automatic conversion that the java compiler makes between primitive types and their corresponding object wrapper classes is autoboxing

Int🡪Integer and vice vera is unboxing.

1. Default values:

boolean – false

char- empty

byte,int,long – 0

double,float 0.0

String, object – null

Constructors:

1.Method that has name Same as class name and no return type. Spl method to initialize object. Invoked by JVM at time of object creation.

2.Two types:

Default constructor & Parameterised constructor

3.Constructor cannot have return type.

4. Possible to throw exception from constructor

5.Any access modifier can be used in constructor.

6.Overload of default cons is possible.

.this and super():

.this:

One constructor can call another constructor via this(parameters) .Why “.this” is first method while calling another constructor? – Java wants to be sure that object was fully constructed.

**this** method cannot be used in other methods.

Super():

JVM first calls super class constructor before sub class. It must be first call in constructor.

Garbage Collection:

The garbage collector manages the allocation and release of memory for an application.

It is a process to identify unused objects and deleting them from heap memory. This process comes into picture when there is no live reference or live thread.

It is controlled by JVM so we don’t know the exact time it runs but it runs when CPU is free /memory is low.

We can request JVM to run the process by System.gc(). But doesnot run immediately

**finalized** method (java.lang.object) is called by GC before collecting any objects

PSVM:

JRE calls main method of Class X.--> X.main(args).

Public🡪JRE need to call this method outside of this class hence it is public.

Static🡪 JRE calls the method with Class name and without an instance of that class.

Void🡪 The program does not return anything so return type is void

Main 🡪 JRE needs to start the program with a method and the name is main. It is just name of the method. Since it is significant, it is called main.

ARRAYS:

Array is a data structure to store more than one value of same data type in memory. they are objects. accessed using their index which Uses Zero based indexing. Object references with fixed size.Always initialized with primitive data type with default values.

Cons:

Array allocate continuous place in the memory even if not used. Length is fixed.

Overcome: create a new array and copy the entire elements to new one by loop or Use System.copyarray. ArrayList or vector can be used which is dynamic.

VarArgs 🡪 Method defined only once with Zero or many arguments.

It can cause performance issue when used a huge number of times.

If pgm uses method 2-3 paramters for 90% pf the time we can use method and use overloading.

So only 10 of the time varargs will be used.

You can not OVERLOAD varAgrs . Compilation error is thrown due to ambiguity

ENCAPSULATION:

Variables are hidden and are accessed only by methods. Data changes should be controlled by metods.

Data hiding.

INHERITANCE:

Why do we override? To define a behavior specific to child class.

Why does it java initialize super class first? Child contains parent class as well. So in order to initialize first parent class needs to be initialized first.

Does a child class inherits the constructor of Super class? NO, it calls them but do not override.

Does java support multiple ineritance? NO. if multiple parent class have same method, java would not know which to inherit. Due to this diamond problem, java doesn’t support.

In java 8, interface came up with default method which is similar to multiple inheritance.

POLYMOPHISM:

Ability of object to take many forms. Two types:

1 Static- during compilation time while method overloading

2.Dynamic – during runtime while overriding

ABSTRACTION:

Hiding implementation from the user. It defines what object does instead of how it does.

Other ques:

DEFAULT METHOD in INTERFACE 🡪 the class need not implement this method.

STATIC EARLY BONDING??

Final,finally,finalize

MARKER INTERFACE – interface with no fields and methods. Eg SERIALIZABLE, CLONEABLE

COLLECTIONS Inetrface:

LIST -> put data orderly in specific way/ control over element inserted/ Access by index/can contain duplicates OCAD

Null can be inserted. Why req? ->Design decision to follow Array functionality.

While adding elements dynamically, it create indexes to the size of the prev count

Capacity vs size of ArrayList – Capacity is the number of elements that can be stored. Size is the totally element currently present in the list. Size <=Capacity. Capacity=length.

How to manage Resizing :

Arraylist create with required capacity.

**Types**:

ArrayList -

LinkedList -

Vector –

**Methods:**

Add (wit index). Get (with index)

indexOf(index)

remove(index)

|  |  |
| --- | --- |
| Array | ArrayList |
| Static – cannot change size | Dynamic – you can resize |
| Work with objects and primitive types | Work with objects and not primitive type |

|  |  |
| --- | --- |
| ArrayList | Vector |
| Non - Synchronised | Synchronised |
| Not thread safe | Thread safe |

ITERARTOR:

Why Iterator instead of loop? Iterator is an interface. Collection interface has the method iterator which return iterator object must implement it.

hasNext()🡪 true if we have more elements,

next()🡪 provides next element

clear() 🡪 removes element if req

Fail fast Principle🡪 System fails when error occurs. Iterarors are by default fail fast

If collection is structurally modified after iterator created, iterator will throw concurrentModificationException.

Fail Safe Princple🡪

System does not fails when error occurs. This is done by cloning collection

It is done by CopyOnWriteArrayList

Set

Queue

MAPS:

HashMap is a map implementation

An object that maps Key to Value.

A key cannot have duplicate. Values can have duplicates

Elements are not kept in order.

Null can be added once in key . Null key will be first while iterating

Many Null values can be added.

NULL key value cannot be added

String is popular in Hashmap as it is immutable. Stored in Stringpool. No risk of change from external party.

Constant time performance is provided by Hashmap -orde pf one is given.and faster than Hastable as not sync.

Methods:

EntrySet()--??

Keyset() 🡪 provides Keys

values() 🡪provides values

getkey, getValue

Same key added🡪 replaces the old key and value

|  |  |
| --- | --- |
| HashMap | HashTable |
| Not sync. Not thread safe. Hence faster | Sync. Thread safe |
| Null allowed in key or value | Null not allowed in key or value |
| Iterator | Enumarator + Iterator |

Does map keep any order?

Hashmap & Hashcode don’t.

LinkedHashmap & Tree map does