Comparisions and Differences between the java concepts

Method - Constructor

Method	Constructor
It is used to perform some operations.	It is used to initialize data members/variables
It will have return type.	It will not have any return type.
We can overload and override it.	We can overload but cannot override it.
It can be classified into static and non-static	It is always non-static.
It can be inherited.	It cannot be inherited.
Method name and class name can be same	Constructor name and class name should always be same
It will be called/invoked by method name.	Whenever an object is created, constructor will get invoked
It can return any value.	It will not return any value.
It should be developed explicitly.	Each and every class will have a constructor(default)

this keyword – super keyword

this keyword	super keyword
this keyword is used to point to the current object.	super keyword is used in the case of method overriding
whenever local and global variable names are same, to differentiate b/w them, we should use this keyword	In the case of method overriding whenever we need super class implementation along with the subclass implementation then we should go for super keyword.
this keyword can be used only in the non-static context because internally this keyword is non-static.	super keyword can be used only in non-static context.

method overloading and method overriding

method overloading	method overriding
Developing multiple methods with same name but variation in the argument list is called as method overloading.	Developing a method in the sub class with same name and signature as in the super class but with different implementation in the sub class is called method overriding.
We can overload both static and non-static.	We can override only non-static.
The method name should be same but there is no restrictions on the access specifier, modifier and return type.	The complete signature should be same in the sub class as in the super class.
It is used to perform some operations.	When we want new/updated implementation we go for method overriding.
There is no rule that it should have “is a relationship”.	It should have “is a relationship” to achieve overriding.
It is an example for compile time polymorphism	It is an example for run time polymorphism.
We can overload main method	We cannot override main method.
We can overload constructor	We cannot override constructor

abstract class - interface

abstract class	interface
Any class which is declared with a keyword abstract is called as abstract class.	interface is a java type which is by default abstract. It is pure abstract body
In abstract class we have 3 members (variables, methods, constructor).	In interface we have 2 members. (variables and methods)
In abstract class we can have constructor	interface doesn't support constructor.
In abstract class, variables can be static and non-static.	In interface, variables are by default static and final.
Methods can have access specifier.	Methods are by default public and abstract
We can develop both abstract and concrete methods in abstract class.	We can develop only abstract methods in interface.
We have to override all the abstract methods in the sub class but not in the concrete class.	All the methods should be overridden.
Abstract class extends object class.	Interface doesn't extend any class.
Through abstract class we can achieve constructor chaining.	Through interface we cannot achieve constructor chaining because it doesn't support constructor
When we know partial implementation then we should go for abstract class	When we don't know 100% implementation the we should go for interface.

Error - Exception

Error	Exception
Error cannot be predicted.	Exception can be predicted.
Errors are occurred due to the system configuration.	Exceptions are occurred due to the mistakes done by the programmer.
Error cannot be handled.	Exception can be handled using try and catch or throws.

static – non static

static	non static
Any member of the class which is declared with the keyword static is called as static member of the class	Any member of the class which is declared without the keyword static is called as non static member of the class
It is always associated with the class	It is always associated with the object
It is always one/single copy	It is always multiple copies
All the static members of the class will get stored inside Static Pool Area(SPA)	All the non static members of the class will get stored inside Heap Memory(HM)
Whenever we want to access static members from one class to another class we have to use classname.variablename or classname.methodname	Whenever we want to access from non static to static we have to create object i.e object.variable_name object.method_name or reference_variable.variable_name reference_variable.method_name
static cannot be overridden	non static can be overridden
method and variable can be static	method and variable can be non static but constructor is always non static

this() calling – super() calling

this()	super()
It is used in the case of constructor overloading	It is used in the case of constructor chaining(inheritance is mandatory)
It is used to call from one constructor to another constructor within the class	It is used to call from one constructor to another constructor between the classes
It should be the first statement inside the constructor	It should be the first statement inside the constructor
We cannot develop 2 this calling statements in the single constructor	We cannot develop 2 super calling statements in the single constructor
It should be written explicitly	It can be written implicitly or explicitly

final – finally – finalize()

final	finally	finalize()
It is a keyword in java	It is a block used in exception handling	It is a non static method which belongs to Object class
It can be used for class and class members except constructor	It will get executed even though the exception is addressed or not	It will get invoke when we write System.gc() and is used for processing garbage collection

Array - ArrayList

Array	ArrayList
It is a linear data structure which is used to store homogeneous type of data	It is a class which belongs to java.util package which should be imported explicitly
To sort an array we should use bubble sorting techniques or Arrays.sort() which belongs java.util package which should be imported explicitly	To sort an ArrayList we should use Collections.sort() which belongs java.util package which should be imported explicitly
It can be single or multi dimensional	It is always single dimensional
size is fixed and it cannot increase its size	size is dynamic and it increases its size by 50%
To find array length we should use array_name.length	To find the size of ArrayList we should use collection_variable.size()
It is faster in performance as its size is fixed	It is slower in performance as its size is dynamic in nature
Assignment operator will be used to store the elements into array	add() will be used to store the elements into ArrayList

Collection - Collections

Collection	Collections
It is a root interface of collection hierarchy which belongs to java.util package which should be imported explicitly	It is a concrete class(utility class) which belongs to java.util package which should be imported explicitly
Since java 8 version, Collection is an interface with static as well as abstract and default methods	Collections operate only with static methods
Here non static methods present are add(),addAll(),peek()....etc	Here methods present are sort(), shuffle(),swap()... etc

HashMap - Hashtable

HashMap	Hashtable
It is being used from JDK 1.2	It is being used from JDK 1.0
It is non synchronized and non thread safe	It is synchronized and is thread safe
Multiple threads can operate simultaneously	Only one thread is allowed to operate Hashtable’s object
Performance is high because of non synchronization	Performance is low because of synchronization
One null key is allowed but multiple null values are allowed	null is not allowed for both key and value
It is traverse by Iterator	It is traverse by Iterator and Enumerator

Collection types

Parameters	ArrayList	Vector	LinkedList	PriorityQueue	HashSet	LinkedHashSet	TreeSet
size	dynamic	dynamic	dynamic	dynamic	dynamic	dynamic	dynamic
increase in size	50%	100%	50%	50%	50%	50%	50%
homo/hetero	both	both	both	only homo	both	both	only homo
allows duplicates	yes	yes	yes	yes	no	no	no
allows null	yes	yes	yes	no	yes	yes	no
follows order of insertion	yes	yes	yes	no	no	yes	no
auto sorting	no	no	no	partially	no	no	completely
indexed type	yes	yes	yes	no	no	no	no
fetch upon index	yes	yes	yes	no	no	no	no
methods	add(Object Obj) add(int index) addAll(Collection c) addAll(int index, Collection c) remove(Object obj) remove(int index) removeAll(Collection c) retainAll(Collection c) size() get(int index) contains(Object obj)	add(Object Obj) add(int index) addAll(Collection c) addAll(int index, Collection c) remove(Object obj) remove(int index) removeAll(Collection c) retainAll(Collection c) size() get(int index) contains(Object obj) capacity()	add(Object Obj) add(int index) addAll(Collection c) addAll(int index, Collection c) remove(Object obj) remove(int index) removeAll(Collection c) retainAll(Collection c) size() get(int index) contains(Object obj) peek() poll()	add(Object Obj) addAll(Collection c) remove(Object obj) removeAll(Collection c) retainAll(Collection c) size() contains(Object obj) peek() poll()	add(Object Obj) addAll(Collection c) remove(Object obj) removeAll(Collection c) retainAll(Collection c) size() contains(Object obj)	add(Object Obj) addAll(Collection c) remove(Object obj) removeAll(Collection c) retainAll(Collection c) size() contains(Object obj)	add(Object Obj) addAll(Collection c) remove(Object obj) removeAll(Collection c) retainAll(Collection c) size() contains(Object obj)

String – StringBuffer - StringBuilder

String	StringBuffer	StringBuilder
String objects will get stored inside String Pool Area(Constant and Non Constant Pool Area)	StringBuffer class objects will get stored inside Heap Memory	StringBuilder class objects will get stored inside Heap Memory
It provides slower performance as it creates separate object on each manipulation	It provides slower performance compared to StringBuilder as it is synchronized and is thread safe	It provides faster performance compared to StringBuffer as it is not synchronized and is not thread safe
String class is immutable	Here strings are mutable	Here strings are mutable

throw – throws - Throwable

throw	throws	Throwable
It is used to create/invoke an exception object of Throwable class type	It is used to propagate an exception from one method to another method	▯ It is the superclass of all errors and exceptions in Java which belongs to java.lang package which will be imported implicitly. ▯ Error and Exception classes will inherit the properties from Throwable class. ▯ It implements Serializable interface
It should be developed inside the method definition	It should be developed/declared in the method declaration	
By using throw keyword, we can throwcreate/invoke only one exception object at a time	By using throws keyword, we can propagate multiple exception objects at a time	

Primitive and non primitive data types

primitive data types	non primitive data types
All the primitive data types in java are predefined	Arrays and Reference type/class type are user defined while String is pre defined
Primitive data structure will contain some value, it cannot have null	non primitive data structure can contain null
Primitive data types help to store values which cannot be referred as Object	non primitive data types help to store values which can be referred as Object
Ex: byte, short, int….etc	Ex: Any class type, String, array

JVM – JRE - JDK

JVM	JRE	JDK
Java Virtual Machine	Java Runtime Environment	Java Development Kit
It is a virtual machine which doesnot exist physically and it is whole responsible to execute the program	It is an environmental setup to run the java program	It is a development kit which consists of all the library files and utilities to develop a java s/w