1. **What does the “static” keyword mean ?**

**The static keyword in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.**

1. **Can you override private or static method in Java ?**

**Static methods also cannot be overridden, because static methods are a part of the Class itself, and not a part of any instance(object) of that class. You however candeclare same static method with same signature in child classes, but that would not be considered as runtime polymorphism (override of methods)**

1. **Can you access non static variable in static context ?**

**Static fields and methods are connected to the class itself and not its instances. If you have a class A, a 'normal' method b and a static method c and make an instance a of your class, the calls to**A.c()**and**a.b()**are valid. Method**c()**has so no idea, which instance is connected, so it cannot use non-static fields.**

1. **What are the Data Types supported by Java ?**

|  |  |
| --- | --- |
| Data Type | Default Value (for fields) |
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| String (or any object) | null |
| boolean | false |

1. **What is Autoboxing and Unboxing ?**

**The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing. This is the new feature of Java5. So java programmer doesn't need to write the conversion code.**

## **Advantage of Autoboxing and Unboxing:**

|  |
| --- |
| No need of conversion between primitives and Wrappers manually so less coding is required. |

1. **Difference between an Interface and an Abstract class ?**

**An**interface**is a description of the behaviour an implementing class will have. The implementing class ensures, that it will have these methods that can be used on it. It is basically a contract or a promise the class has to make.**

An abstract class is a basis for different subclasses that share behaviour which does not need to be repeatedly created. Subclasses must complete the behaviour and have the option to override predefined behaviour (as long as it is not defined as final or private ).

1. **What is the difference between pass by reference and pass by value ?**

**It's a way how to pass arguments to functions. Passing by reference means the called functions' parameter will be the same as the callers' passed argument (not thevalue, but the identity - the variable itself). Pass by value means the called functions' parameter will be a copy of the callers' passed argument.**

1. **What are the basic interfaces of Java Collections Framework ?**

[**The Collection Interface**](https://docs.oracle.com/javase/tutorial/collections/interfaces/collection.html)

**The Set Interface**

**The List Interface**

**The Queue Interface**

**The Deque Interface**

**The Map Interface**

**The SortedSet Interface**

**The SortedMap Interface**

1. **Why do we use an Iterator while using Java Collections ?**

**For example, you might want to display each element. The easiest way to do this is to employ an iterator, which is an object that implements either the Iterator or the ListIterator interface. Iterator enables you to cycle through a collection, obtaining or removing elements.**

1. **What is the difference between HashMap and Hashtable ?**

|  |  |
| --- | --- |
| HashMap | Hashtable |
| 1) HashMap is non synchronized. It is not-thread safe and can't be shared between many threads without proper synchronization code. | Hashtable is synchronized. It is thread-safe and can be shared with many threads. |
| 2) HashMap allows one null key and multiple null values. | Hashtable doesn't allow any null key or value. |
| 3) HashMap is a new class introduced in JDK 1.2. | Hashtable is a legacy class. |
| 4) HashMap is fast. | Hashtable is slow. |
| 5) We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap); | Hashtable is internally synchronized and can't be unsynchronized. |
| 6) HashMap is traversed by Iterator. | Hashtable is traversed by Enumerator and Iterator. |
| 7) Iterator in HashMap is fail-fast. | Enumerator in Hashtable is not fail-fast. |
| 8) HashMap inherits AbstractMap class. | Hashtable inherits Dictionary class. |

1. **What is difference between Array and ArrayList ? When will you use Array over ArrayList ?**

**ArrayList and LinkedList are two popular collection classes in Java and Majordifference between ArrayList and LinkedList is on there implementation whileArrayList uses index based Array LinkedList is implemented using Doubly Linked List Data Structure**

**We can use Array when we the size of the list is fixed.**

1. **When to use parseInt method?**

**Type casting is used to make object of one type assign to other without altering or parsing the contents of the object. While **parseInt** method parses the String values and converts it to integer value i.e. converts String value to integer.**

1. **What is the difference between StringBuffer and StringBuilder class?**

**StringBuffer and StringBuilder have the same methods with one difference and that's of synchronization. StringBuffer is synchronized( which means it is thread safe and hence you can use it when you implement threads for your methods) whereas StringBuilder is not synchronized( which implies it isn't thread safe).**

1. **What is finalize method?**

**Every class inherits the finalize() method from java.lang.Object.**

The method is called by the garbage collector when it determines no more references to the object exist.

The Object finalize method performs no actions but it may be overridden by any class.

1. **What is the difference between final, finally and finalize method?**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | final | finally | finalize |
| 1) | Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed. | Finally is used to place important code, it will be executed whether exception is handled or not. | Finalize is used to perform clean up processing just before object is garbage collected. |
| 2) | Final is a keyword. | Finally is a block. | Finalize is a method. |

1. **What is the difference between throw and throws keyword?**

**throw keyword is used to throw Exception from any method or static block in Java while throws keyword, used in method declaration, denoted which Exception can possible be thrown by this method. They are not interchangeable.**

1. **When and why to use “super” keyword ?**

****Usage of java super Keyword****

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.
4. **What is “jar” file and how to create and use it?**

**A **JAR** (Java ARchive) is a package **file** format typically used to aggregate many Java class **files** and associated metadata and resources (text, images, etc.) into one**file** for distribution.**

****Create the JAR file.** The format of the command line for creating the JAR file looks like this: "jar cf 'jar-file'.jar input-file(s)."**

* The "jar" portion refers to the jar.exe program, which compiles the JAR file.
* The "c" option specifies that you want to create a JAR file
* The "f" option means that you want to specify the filename.
* The "jar-file" portion is where you should type the name that you want the file to have.
* "Input-file(s)" is a space-separated list of all the files to be included in the JAR file.
* For example, you might type "jar cf myjar manifest.txt myclass.class." This would create a JAR file with the filename "myjar.jar" which would include the files "manifest.txt" and "myclass.class."
* If you add directories to the JAR file, the jar.exe utility will automatically add their contents.

java.exe -jar file.jar

1. **What is the purpose of fileInputStream and fileReader class?**

fileReader**reads char by char**

fileInputStream reads byte by byte

1. **What is the purpose of fileOutputStream and fileWriter class?**

**A**FileOutputStream**writes bytes directly. A**FileWriter**encapsulates a**FileOutputStream**(by creating it in the**FileWriter**constructor as in your question) and provides convenience methods to write characters and Strings.**

**All the programs should be written in Java Language.**

1. **Write a program to implement Binary Search?**

import java.util.Scanner;

class BinarySearch

{

public static void main(String args[])

{

int c, first, last, middle, n, search, array[];

Scanner in = new Scanner(System.in);

System.out.println("Enter number of elements");

n = in.nextInt();

array = new int[n];

System.out.println("Enter " + n + " integers");

for (c = 0; c < n; c++)

array[c] = in.nextInt();

System.out.println("Enter value to find");

search = in.nextInt();

first = 0;

last = n - 1;

middle = (first + last)/2;

while( first <= last )

{

if ( array[middle] < search )

first = middle + 1;

else if ( array[middle] == search )

{

System.out.println(search + " found at location " + (middle + 1) + ".");

break;

}

else

last = middle - 1;

middle = (first + last)/2;

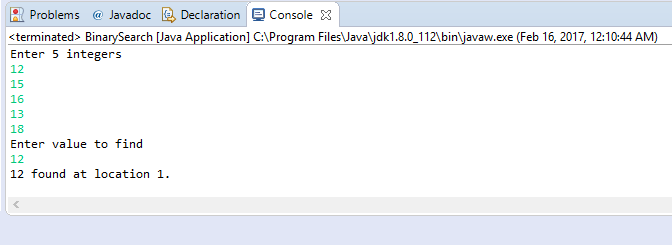
}

if ( first > last )

System.out.println(search + " is not present in the list.\n");

}

}



1. **Write a program to implement Bubble sort?**

import java.util.Scanner;

class BubbleSort {

public static void main(String []args) {

int n, c, d, swap;

Scanner in = new Scanner(System.*in*);

System.*out*.println("Input number of integers to sort");

n = in.nextInt();

int array[] = new int[n];

System.*out*.println("Enter " + n + " integers");

for (c = 0; c < n; c++)

array[c] = in.nextInt();

for (c = 0; c < ( n - 1 ); c++) {

for (d = 0; d < n - c - 1; d++) {

if (array[d] > array[d+1]) /\* For descending order use < \*/

{

swap = array[d];

array[d] = array[d+1];

array[d+1] = swap;

}

}

}

System.*out*.println("Sorted list of numbers");

for (c = 0; c < n; c++)

System.*out*.println(array[c]);

in.close();

}

}

****

1. **Write a program to implement the stack operations?**

public class MyStack {

private int maxSize;

private long[] stackArray;

private int top;

public MyStack(int s) {

maxSize = s;

stackArray = new long[maxSize];

top = -1;

}

public void push(long j) {

stackArray[++top] = j;

}

public long pop() {

return stackArray[top--];

}

public long peek() {

return stackArray[top];

}

public boolean isEmpty() {

return (top == -1);

}

public boolean isFull() {

return (top == maxSize - 1);

}

public static void main(String[] args) {

MyStack theStack = new MyStack(10);

theStack.push(10);

theStack.push(20);

theStack.push(30);

theStack.push(40);

theStack.push(50);

while (!theStack.isEmpty()) {

long value = theStack.pop();

System.*out*.print(value);

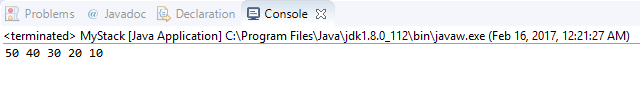
System.*out*.print(" ");

}

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

}

}

****

1. **Write a program to search any data in given LinkList without using collections framework?**

**package CoreXJava;**

**public** **class** ArraySearch

{

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

{

**int**[ ] numbers = { 12, 13, 2, 33, 23, 31, 22, 6, 87, 16 };

**int** key = 31;

**int** i = 0;

**boolean** found = **false**; // set the boolean value to false until the key is found

**for** ( i = 0; i < numbers.length; i++)

{

**if** (numbers[ i ] == key)

{

found = **true**;

**break**;

}

}

**if** (found) //When found is true, the index of the location of key will be printed.

{

System.***out***.println("Found " + key + " at index " + i + ".");

}

**else**

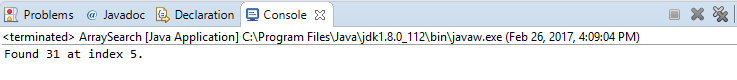
{

System.***out***.println(key + "is not in this array.");

}

}

**}**

****

1. **Write a program to Add and Delete data inside LinkList without using collections framework?**

**package** CoreXJava;

**import** java.util.Scanner;

**public** **class** Array1

{

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

{

**int** n, x, flag = 1, loc = 0;

Scanner s = **new** Scanner(System.***in***);

System.***out***.print("Enter no. of elements you want in array:");

n = s.nextInt();

**int** a[] = **new** **int**[n];

System.***out***.println("Enter all the elements:");

**for** (**int** i = 0; i < n; i++)

{

a[i] = s.nextInt();

}

System.***out***.print("Enter the element you want to delete:");

x = s.nextInt();

**for** (**int** i = 0; i < n; i++)

{

**if**(a[i] == x)

{

flag =1;

loc = i;

**break**;

}

**else**

{

flag = 0;

}

}

**if**(flag == 1)

{

**for**(**int** i = loc+1; i < n; i++)

{

a[i-1] = a[i];

}

System.***out***.print("After Deleting:");

**for** (**int** i = 0; i < n-2; i++)

{

System.***out***.print(a[i]+",");

}

System.***out***.print(a[n-2]);

}

**else**

{

System.***out***.println("Element not found");

}

s.close();

}

}

