```
package com.journaldev.files;

import java.io.File;

public class RenameFileJava {

    /**
     * Rename File  or Move File in Java example
     * @param args
     */
    public static void main(String[] args) {
        //absolute path rename file
        File file = new File("/Users/pankaj/java.txt");
        File newFile = new File("/Users/pankaj/java1.txt");
        if(file.renameTo(newFile)){
            System.out.println("File rename success");;
        }else{
            System.out.println("File rename failed");
        }

        //relative path rename file
        file = new File("DB.properties");
```

We should always check the renameTo return value to make sure rename file is successful because it's platform dependent and it doesn't throw IO exception if rename fails. That's all for a quick java rename file and move file example.

```
    }

    //relative path rename file
    file = new File("DB.properties");
    newFile = new File("DB_New.properties");
    if(file.renameTo(newFile)){
        System.out.println("File rename success");;
    }else{
        System.out.println("File rename failed");
    }

    //java move file from one directory to another
    file = new File("/Users/pankaj/DB.properties");
    newFile = new File("DB_Move.properties");
    if(file.renameTo(newFile)){
        System.out.println("File move success");;
    }else{
        System.out.println("File move failed");
    }

    //when source file is not present
    file = new File("/Users/pankaj/xyz.txt");
```

We should always check the renameTo return value to make sure rename file is successful because it's platform dependent and it doesn't throw IO exception if rename fails. That's all for a quick java rename file and move file example.

```
            System.out.println("File move failed");
        }

        //when source file is not present
        file = new File("/Users/pankaj/xyz.txt");
        newFile = new File("xyz.properties");
        if(file.renameTo(newFile)){
            System.out.println("File move success");;
        }else{
            System.out.println("File move failed");
        }

        // when destination already have a file with same name
        file = new File("/Users/pankaj/export.sql");
        newFile = new File("/Users/pankaj/java1.txt");
        if(file.renameTo(newFile)){
            System.out.println("File move success");;
        }else{
            System.out.println("File move failed");
        }
    }
```

We should always check the renameTo return value to make sure rename file is successful because it's platform dependent and it doesn't throw IO exception if rename fails. That's all for a quick java rename file and move file example.

Java Tricky Interview Question 1 Answer with Explanation

As we know that we can assign null to any object, so doesn't compiler complains about this program? According to java specs, in case of overloading, compiler picks the *most specific function*. Obviously String class is more specific than Object class, hence it will print "String impl".
What if we have another method in the class like below:


```
public static void foo(StringBuffer i){
                System.out.println("StringBuffer impl");
        }
```

In this case, java compiler will throw an error as "The method foo(String) is ambiguous for the type Test" because String and StringBuffer, none of them are more specific to others. A method is more specific than another if any invocation handled by the first method could be passed on to the other one without a compile-time type error. We can pass String as a parameter to Object argument and String argument but not to StringBuffer argument method.

In this case, java compiler will throw an error as "The method foo(String) is ambiguous for the type Test" because String and StringBuffer, none of them are more specific to others. A method is more specific than another if any invocation handled by the first method could be passed on to the other one without a compile-time type error. We can pass String as a parameter to Object argument and String argument but not to StringBuffer argument method.

Java Tricky Programming Question 2 Answer with Explanation

The output of the code snippet will be:

```
31536000000
1471228928
```

In case of the first variable, we are explicitly creating it as long by placing an "L" at the end, so the compiler will treat this at long and assign it to the first variable.

In the second case, the compiler will do the calculation and treat it as a 32-bit integer, since the output is outside the range of integer max value (2147483647), the compiler will truncate the most significant bits and then assign it to the variable.

Binary equivalent of 1000*60*60*24*365L = 0111010101111011000100101110000000000 (36 bits)
Removing 4 most significant bits to accommodate in 32-bit int, value = 01010111101100010010110000000000 (32 bits)
Which is equal to 1471228928 and hence the output.

Recently I have created YouTube videos for java tricky programs, you should check them out. Also subscribe to my YouTube Channel to get notified when I add new videos.

Suraj Vishwakarma says

JUNE 9, 2017 AT 12:29 PM

ye the output of this program is true

```
public class conditionalstate
{
public static void main(String args[])
{
boolean a = false; //without any change
if(a=true)
{
System.out.println("a is true");
}
else
{
System.out.println("a is false");
}
}
}
```

o/p

a is true

Reply

Bishnu says

MARCH 16, 2018 AT 12:43 AM

"a=true" is an assignment operation and not a relational operator. "a=true", would return the value as true. Hence, you are getting a is true.

Reply

Java Tricky Interview Question 1

What is the output of the below program?

```
public class Test {
	public static void main(String[] args) {
		foo(null);
	}
	public static void foo(Object o) {
		System.out.println("Object impl");
	}
	public static void foo(String s) {
		System.out.println("String impl");
	}
}
```

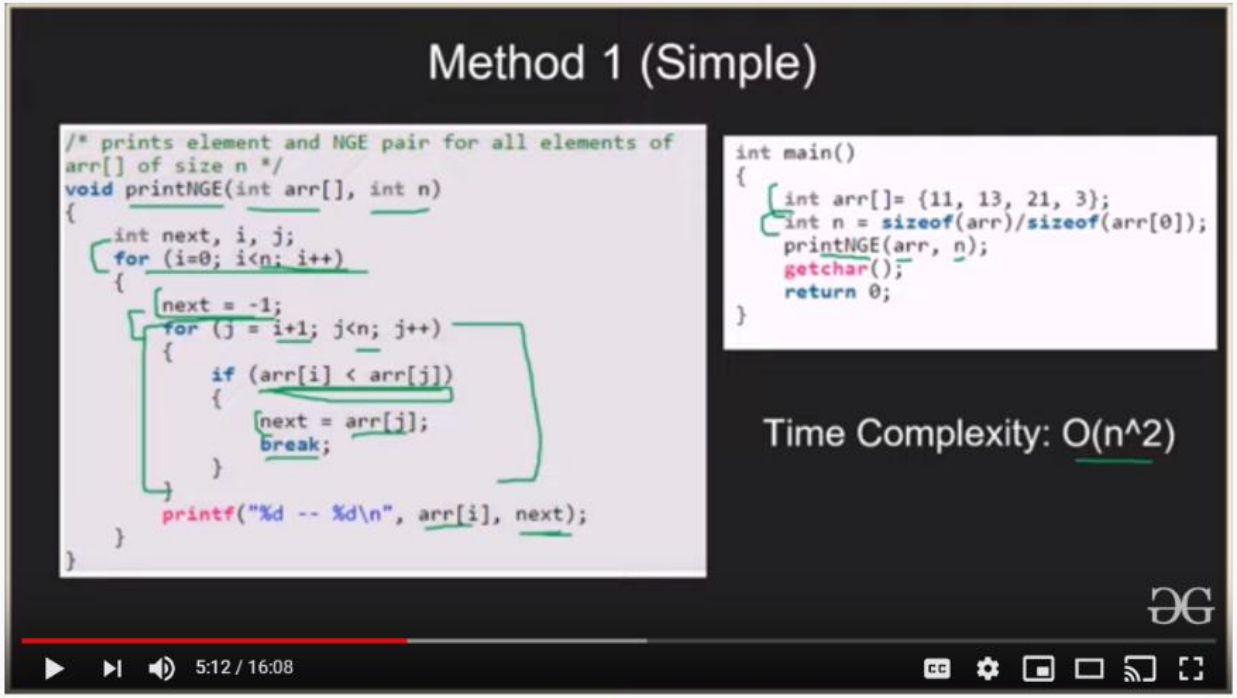
Java Tricky Programming Question 2

What will below statements print?

```
long longWithL = 1000*60*60*24*365L;
long longWithoutL = 1000*60*60*24*365;
System.out.println(longWithL);
System.out.println(longWithoutL);
```

****

print closet element in array



Next Greater Element | GeeksforGeeks

36,091 views • Oct 30, 2016

94 21 SHARE SAVE

GeeksforGeeks

SUBSCRIBE

Up next AUTOPLAY

next greater element (use of stack)

Vivekanand Khyade - Algorithm E...

22K views

How to attempt GATE 2020 | Last Minute GATE Exam Tips |...

Gate Lectures by Ravindraba...

Recommended for you

New

Dynamic Programming | Set 11 (Egg Dropping Puzzle) |...

GeeksforGeeks

78K views

The best software Interview material - Prepare in less than ...

Gaurav Sen

235K views

Program #1: Write a program to print hello world without using a semicolon anywhere in the code in java

```
01. package instanceofjava;
02. /**
03.  * Print semicolon without using semicolon in java
04.  * Print hello world without using semicolon java program code
05.  * @author www.instanceofjava.com
06.  */
07. public class PrintSemiColon {
08.
09. public static void main(String[] args) {
10.
11.    if(System.out.printf("Hello World")!=null){
12.
13.    }
14.
15.
16. }
17. }
```

Output:

```
01. Hello World
```

```
1  package instanceofjava;
2  /**
3   * Print semicolon without using semicolon in java
4   * Print hello world without using semicolon java program code
5   * @author www.instanceofjava.com
6   */
7  public class PrintSemiColon {
8
9  public static void main(String[] args) {
10
11     if(System.out.printf("Hello World")!=null){
12
13     }
14
15
16 }
17 }
18
```

Print semicolon without using semicolon in java

Posted by: InstanceOfJava Posted date: Jun 10, 2017 / comment : 0

- Write a program to print a semicolon without using a semicolon anywhere in the code in java.
- Write a program to print"hello world"without using semicolon anywhere in the code.\
- Yes we can print ";" without using semicolon in java by using printf in java.
- Format text using printf in java here we have provided information about how to format text using printf in java.
- In the same way we can print ";" using printf .
- ASCII value for ";" is 59. So if we print 59 in character format it will print ;
- System.out.printf("%C",59);
- But in java program end of every statement we need to keep ";". To eliminate this we can use if condition.
- Keep print statement inside if condition and it will print the ";" now.
- Now let us see an example java program to print a semicolon without using a semicolon anywhere in the code in java.

Return statement in try catch block java

Posted by: InstanceOfJava Posted date: Mar 15, 2016 / comment : 5

1.Can we write return statement in try or catch blocks in java

- Inside method if we have some statements which may proven to raise exceptions we need to keep those statements in side try catch blocks in order to handle the exceptions.
- There is a situation where a method will have a return type and we can return some value at any part of the method based on the conditions.
- So mixing these two scenarios we may have a situation to return some value from try block or catch block in such cases we need to follow some basic rules.
- This is also one of the famous interview question "can we write return statement in try block ", "can we write return statement in catch block ", "what will happen if we return some value in try block or catch block ", "will catch block execute after return statement " and "what are the basic rules we need to follow when we are returning or our method returning some value in try or catch blocks"
- We will see the all the scenarios and conclusion.

Return statement in try block:

i) return statement in try block only

- If we return a value in try block and if we do not return a value at the end of the method it leads to compile time exception

i) return statement in try block only

- If we return a value in try block and if we do not return a value at the end of the method it leads to compile time exception
- Error: This method must return a result of type int

```
01. package com.exceptionhandlingiinterviewquestions;
02.
03. public class TryCatchReturn{
04.
05. int calc(){ // Error:This method must return a result of type int
06.
07. try {
08.
09. return 1;
10.
11. } catch (Exception e) {
12.
13. }
14.
15.  System.out.println("End of the method");
16. }
17.
18.
19. public static void main(String[] args) {
20.
21.       TryCatchReturn obj = new TryCatchReturn();
22.
23.
24. }
25.
26. }
```

ii) return statement in try block and end of the method but after return one statement

- Even though it in this condition it wont reach end of the method still it asks us for return at end of the method because in some cases there may be a chance of getting exception in try block so it will not completely execute try in this case we don not have return in catch so at end of the method it expects some value to be return because the method have some return type.

ii) return statement in try block and end of the method but after return one statement

- Even though it in this condition it wont reach end of the method still it asks us for return at end of the method because in some cases there may be a chance of getting exception in try block so it will not completely execute try in this case we don not have return in catch so at end of the method it expects some value to be return because the method have some return type.

- If we are keeping return statement in try block only there may be a situation of chance of raising exception and try will not execute completely and it goes to catch block that is the reason it expecting a return at catch or end of the method
- So in this scenario we will try to keep return but after return we have written one statement

```
01. package com.exceptionhandlingiinterviewquestions;
02.
03. public class TryCatchReturn{
04.
05. int calc(){
06.
07. try {
08.
09. return 1;
10.
11. } catch (Exception e) {
12.
13. }
14.
15. return 10;
16.  System.out.println("End of the method"); // Error : Unreachable code
17. }
18.
19.
```

iii) return statement in try block and end of the method

- This is the correct and successful scenario with respect to try with return statement in java
- Lest see a java program on this


```
01. package com.exceptionhandlingiinterviewquestions;
02.
03. public class TryCatchReturn{
04.
05. int calc(){
06.
07. try {
08.
09. return 1;
10.
11. } catch (Exception e) {
12.
13. }
14.
15. System.out.println("End of the method");
16. return 10;
17. }
18.
19.
20. public static void main(String[] args) {
21.
22.
23. TryCatchReturn obj = new TryCatchReturn();
24.
25. System.out.println(obj.calc());
26.
27. }
28.
29. }
```

Output:

```
01. 1
```

```
26.
27. }
```

- in the above program last statement will not execute at any condition so it became unreachable code as we know java does not supports unreachable codes so it will raise compile time error.

ii) return statement in catch block and end of the method

```
01. package com.exceptionhandlingiinterviewquestions;
02.
03. public class TryCatchReturn{
04.
05. int calc(){
06.
07. try {
08.
09.  int x=12/0;
10. } catch (Exception e) {
11.
12. return 1;
13. }
14.
15. return 10;
16. }
17.
18.
19. public static void main(String[] args) {
20.
21. TryCatchReturn obj = new TryCatchReturn();
22.
23.  System.out.println(obj.calc());
24.
25. }
26.
27. }
```

Output:

```
01. 1
```

Return statement with try catch block

```
26. }
27. }
```

Output:

```
01. 1
```

Return statement with try catch block

```
1  package trycatchwithreturn;
2
3  public class TryCatchReturn {
4
5      int calc(){
6
7          try {
8              return 10;
9
10         } catch (Exception e) {
11             return 20;
12         }
13
14         return 10;
15     }
16
17 public s                    tring[] args) {
18
19     TryC                   new TryCatchReturn();
20
21     System.out.println(obj.calc());
22
23 }
24 }
25
```

Unreachable code
1 quick fix available:
Remove

return statement allow in try ,catch then not in method.


```
1  package trycatchwithreturn;
2
3  public class TryCatchReturn {
4
5      int calc(){
6
7          try {
8              return 10;
```


```
package trycatchwithreturn;

public class TryCatchReturn {

    int calc(){

        try {
            return 10;

        } catch (Exception e) {
            return 20;
        }

    }

  public static void main(String[] args) {

    TryCatchReturn obj = new TryCatchReturn();

    System.out.println(obj.calc());

}
}
```

```java
public class LinkedListTest2 {
    public static void main(String[] args)
    {
        // Declaring linkedlist without any initial size
        LinkedList<Integer> linkedli = new LinkedList<Integer>();

        // Appending elements at the end of the list
        linkedli.add(new Integer(1));
        linkedli.add(new Integer(2));
        linkedli.add(new Integer(3));
        linkedli.add(new Integer(4));
        linkedli.add(new Integer(5));
        System.out.print("Elements before reversing: " + linkedli);

        // Calling user defined function for reversing
        linkedli = reverseLinkedList(linkedli);
        System.out.print("\nElements after reversing: " + linkedli);
    }

    // Takes a linkedlist as a parameter and returns a reversed linkedlist
    public static LinkedList<Integer> reverseLinkedList(LinkedList<Integer> llist)
    {
        for (int i = 0; i < llist.size() / 2; i++) {
            Integer temp = llist.get(i);
            llist.set(i, llist.get(llist.size() - i - 1));
            llist.set(llist.size() - i - 1, temp);
        }

        // Return the reversed arraylist
        return llist;
    }
}
```

Time Complexity: O(n/2)

Space Complexity: O(1)

returns the reversed list.

NOTE: Collections.reverse() method uses the same algorithm as "By writing our own function(Without using additional space)"

```java
// Java program for reversing a linked list using
// In-built collections class
import java.util.*;

public class LinkedListTest3 {
    public static void main(String[] args)
    {
        // Declaring linkedlist without any initial size
        LinkedList<Integer> linkedli = new LinkedList<Integer>();

        // Appending elements at the end of the list
        linkedli.add(new Integer(1));
        linkedli.add(new Integer(2));
        linkedli.add(new Integer(3));
        linkedli.add(new Integer(4));
        linkedli.add(new Integer(5));
        System.out.print("Elements before reversing: " + linkedli);

        // Collections.reverse method takes a list as a
        // parameter and returns the reversed list
        Collections.reverse(linkedli);

        System.out.print("\nElements after reversing: " + linkedli);
    }
}
```

Time Complexity: O(n/2)

Space Complexity: O(1)

```java
// Java program for reversing linked list using additional space
import java.util.*;

public class LinkedListTest1 {
    public static void main(String[] args)
    {
        // Declaring linkedlist without any initial size
        LinkedList<String> linkedli = new LinkedList<String>();
        // Appending elements at the end of the list
        linkedli.add("Cherry");
        linkedli.add("Chennai");
        linkedli.add("Bullet");
        System.out.print("Elements before reversing: " + linkedli);
        linkedli = reverseLinkedList(linkedli);
        System.out.print("\nElements after reversing: " + linkedli);
    }

    // Takes a linkedlist as a parameter and returns a reversed linkedlist
    public static LinkedList<String> reverseLinkedList(LinkedList<String> llist)
    {
        LinkedList<String> revLinkedList = new LinkedList<String>();
        for (int i = llist.size() - 1; i >= 0; i--) {

            // Append the elements in reverse order
            revLinkedList.add(llist.get(i));
        }
        // Return the reversed arraylist
        return revLinkedList;
    }
}
```

Time Complexity: O(n)

Space Complexity: O(n)

4.Reversing a linked list of user defined objects: An Employee class is created for creating user defined objects with employeeID, employeeName, departmentName as class variables which are initialized in the constructor. An linked list is created that takes only Employee(user defined) Objects. These objects are added to the linked list using add() method. The linked list is reversed using In-built reverse() method of Collections class.

printElements() method is used to iterate through all the user defined objects in the linked list and print the employee ID, name and department name for every object.

```
// Java program for reversing a inkedlist of user defined objects
import java.util.*;

class Employee {
    int empID;
    String empName;
    String deptName;

    // Constructor for initializing the class variables
    public Employee(int empID, String empName, String deptName)
    {
        this.empID = empID;
        this.empName = empName;
        this.deptName = deptName;
    }
}

public class LinkedListTest4 {
    public static void main(String[] args)
    {
        // Declaring linkedList without any initial size
        LinkedList<Employee> linkedli = new LinkedList<Employee>();
```

```
        // Creating user defined objects
        Employee emp1 = new Employee(123, "Cherry", "Fashionist");
        Employee emp2 = new Employee(124, "muppala", "Development");
        Employee emp3 = new Employee(125, "Bullet", "Police");

        // Appending all the objects to linkedList
        linkedli.add(emp1);
        linkedli.add(emp2);
        linkedli.add(emp3);
        System.out.print("Elements before reversing: ");
        printElements(linkedli);

        // Collections.reverse method takes a list as a parameter
        // and returns the reversed list
        Collections.reverse(linkedli);

        System.out.print("\nElements after reversing: ");
        printElements(linkedli);
    }

    // Iterate through all the elements and print
    public static void printElements(LinkedList<Employee> llist)
    {
        for (int i = 0; i < llist.size(); i++) {
            System.out.print("\n EmpID:" + llist.get(i).empID + ", EmpName:"
                             + llist.get(i).empName + ", Department:" + llist.get(i
        }
    }
}
```

Time Complexity: O(n/2)

Space Complexity: O(1)

Output:

9

Scanner.skip skips a input which matches the pattern, here the pattern is :-

(\r\n|[\n\r\u2028\u2029\u0085])?

- ? matches exactly zero or one of the previous character.
- | Alternative
- [] Matches single character present in
- \r matches a carriage return
- \n newline
- \u2028 matches the character with index 2018 base 16(8232 base 10 or 20050 base 8) case sensitive
- \u2029 matches the character with index 2029 base 16(8233 base 10 or 20051 base 8) case sensitive
- \u0085 matches the character with index 85 base 16(133 base 10 or 205 base 8) case sensitive

1st Alternative \r\n

- \r matches a carriage return (ASCII 13)
- \n matches a line-feed (newline) character (ASCII 10)

2nd Alternative [\n\r\u2028\u2029\u0085]

- Match a single character present in the list below [\n\r\u2028\u2029\u0085]
- \n matches a line-feed (newline) character (ASCII 10)
- \r matches a carriage return (ASCII 13)

Linked

2 What does scanner.sl
[\n\r\u2028\u2029\u00

1st Alternative \r\n

- \r matches a carriage return (ASCII 13)
- \n matches a line-feed (newline) character (ASCII 10)

2nd Alternative [\n\r\u2028\u2029\u0085]

- Match a single character present in the list below [\n\r\u2028\u2029\u0085]
- \n matches a line-feed (newline) character (ASCII 10)
- \r matches a carriage return (ASCII 13)
- \u2028 matches the character with index 202816 (823210 or 200508) literally (case sensitive) LINE SEPARATOR
- \u2029 matches the character with index 202916 (823310 or 200518) literally (case sensitive) PARAGRAPH SEPARATOR
- \u0085 matches the character with index 8516 (13310 or 2058) literally (case sensitive) NEXT LINE

share improve this answer

edited Aug 31 '18 at 9:02

answered Aug 31 '18 at 8:15

Surya Prakash Singh

124 • 5

add a comment

Java program to check whether a string is a Palindrome

Introduction to Spring Boot

Print characters and their frequencies in order of occurrence using a LinkedHashMap in Java

What are the main differences between the Java platform and other platforms?

Count of words ending at the given suffix in Java

How to convert an Array to String in Java?

What is Slipped Condition in Multi-threading?

Go vs Java

Find the count of M character words which have at least one character repeated

Add the given digit to a number

```java
/* Java program to demonstrate how to implement static and non-static
   classes in a java program. */
class OuterClass{
   private static String msg = "GeeksForGeeks";

   // Static nested class
   public static class NestedStaticClass{

       // Only static members of Outer class is directly accessible in nested
       // static class
       public void printMessage() {

         // Try making 'message' a non-static variable, there will be
         // compiler error
         System.out.println("Message from nested static class: " + msg);
       }
    }

    // non-static nested class - also called Inner class
    public class InnerClass{

       // Both static and non-static members of Outer class are accessible in
       // this Inner class
       public void display(){
          System.out.println("Message from non-static nested class: "+ msg);
       }
    }
}
class Main
{
    // How to create instance of static and non static nested class?
    public static void main(String args[]){

       // create instance of nested Static class
       OuterClass.NestedStaticClass printer = new OuterClass.NestedStaticClass();

       // call non static method of nested static class
       printer.printMessage();
```

threading?

Go vs Java

Find the count of M character words which have at least one character repeated

Add the given digit to a number stored in a linked list using recursion

How String Hashcode value is calculated?

Java.util.function.IntPredicate interface in Java with Examples

Java.util.function.BiPredicate interface in Java with Examples

How to convert a String to an Int in Java?

```
    }
}
class Main
{
    // How to create instance of static and non static nested class?
    public static void main(String args[]){

        // create instance of nested Static class
        OuterClass.NestedStaticClass printer = new OuterClass.NestedStaticClass();

        // call non static method of nested static class
        printer.printMessage();

        // In order to create instance of Inner class we need an Outer class
        // instance. Let us create Outer class instance for creating
        // non-static nested class
        OuterClass outer = new OuterClass();
        OuterClass.InnerClass inner  = outer.new InnerClass();

        // calling non-static method of Inner class
        inner.display();

        // we can also combine above steps in one step to create instance of
        // Inner class
        OuterClass.InnerClass innerObject = new OuterClass().new InnerClass();

        // similarly we can now call Inner class method
        innerObject.display();
    }
}
```

Output:

```
Message from nested static class: GeeksForGeeks
Message from non-static nested class: GeeksForGeeks
Message from non-static nested class: GeeksForGeeks
```


Most popular in Java

CharBuffer limit() methods in Java with Exam

Path getName(int) method in Java with Examples

What are the differences between static and non-static nested classes?

Following are major differences between static nested class and non-static nested class. Non-static nested class is also called Inner Class.

1) Nested static class doesn't need reference of Outer class, but Non-static nested class or Inner class requires Outer class reference.

2) Inner class(or non-static nested class) can access both static and non-static members of Outer class. A static class cannot access non-static members of the Outer class. It can access only static members of Outer class.

3) An instance of Inner class cannot be created without an instance of outer class and an Inner class can reference data and methods defined in Outer class in which it nests, so we don't need to pass reference of an object to the constructor of the Inner class. For this reason Inner classes can make program simple and concise.

```
/* Java program to demonstrate how to implement static and non-static
   classes in a java program. */
class OuterClass{
    private static String msg = "GeeksForGeeks";

    // Static nested class
    public static class NestedStaticClass{

        // Only static members of Outer class is directly accessible in nested
        // static class
        public void printMessage() {

          // Try making 'message' a non-static variable, there will be
          // compiler error
          System.out.println("Message from nested static class: " + msg);
        }
    }

    // non-static nested class - also called Inner class
```

questions - design pattern
- interview questions - spring
- interview questions - hibernate
- interview questions - core java
- interview questions - arraylist
- interview questions - java enum
- interview questions - java swing
- interview questions - java common
- interview questions - support
- interview questions - technical
- interview questions - java main
- interview questions - hashmap
- interview

What will happen if you put return statement or System.exit () on try or catch block? Will finally block execute?

This is a very popular tricky Java question and it's tricky because many programmers think that no matter what, but the finally block will always execute. This question challenge that concept by putting a return statement in the try or catch block or calling `System.exit()` from try or catch block. Answer of this tricky question in Java is that finally block will execute even if you put a return statement in the try block or catch block but finally block won't run if you call `System.exit()` from try or catch block.

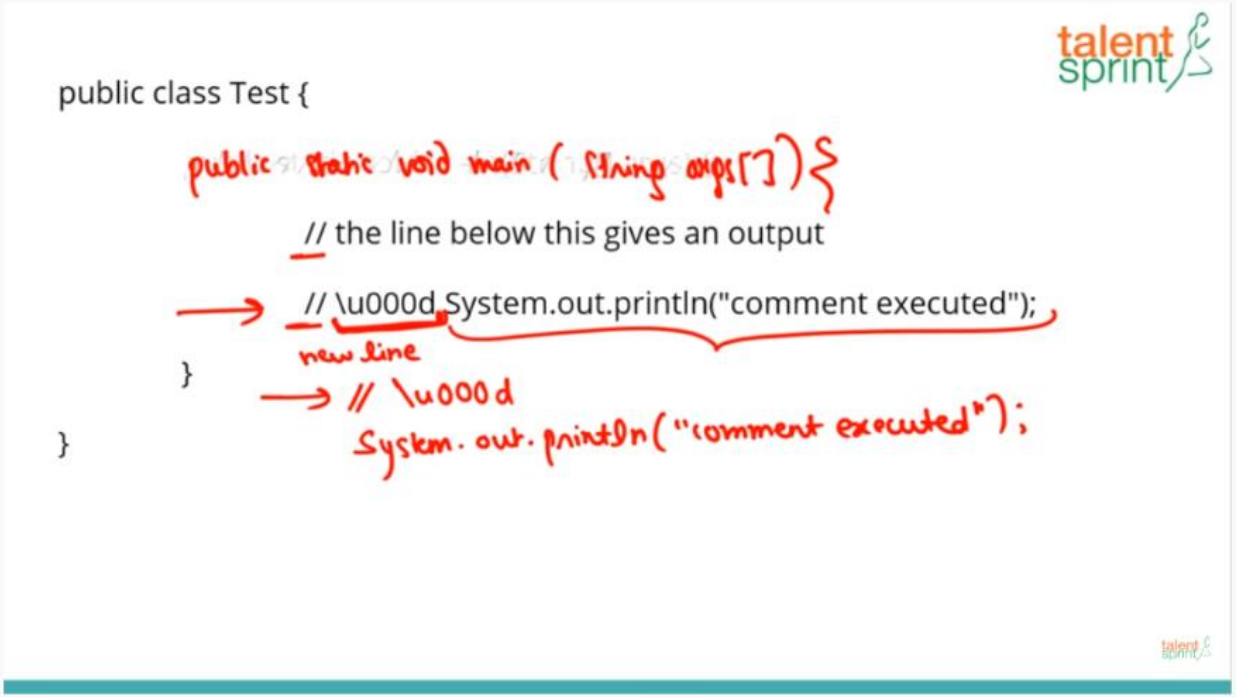
Question: Can you override a private or static method in Java?

Another popular Java tricky question, As I said method overriding is a good topic to ask trick questions in Java. Anyway, you can not override a private or static method in Java, if you create a similar method with same return type and same method arguments in child class then it will hide the superclass method, this is known as method hiding.

Similarly, you cannot override a private method in sub class because it's not accessible there, what you

Books and Resources

- Best Book to Learn Java in 2017
- 5 Books to Learn Spring MVC and Core in 2017
- 2 books to learn Hibernate in 2017
- 12 Advanced Java Programming Books for Experienced Programmers
- 5 Free JavaScript Books to download
- 5 Books to Improve Your Coding Skill
- Books Every Programmer Should Read
- Top 10 Computer Algorithm Books
- 10 Free Java Programming Books
- 5 Books to Learn Java 8 Better
- Books Every Programmer Should Read
- Top 10 Computer Algorithm Books



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

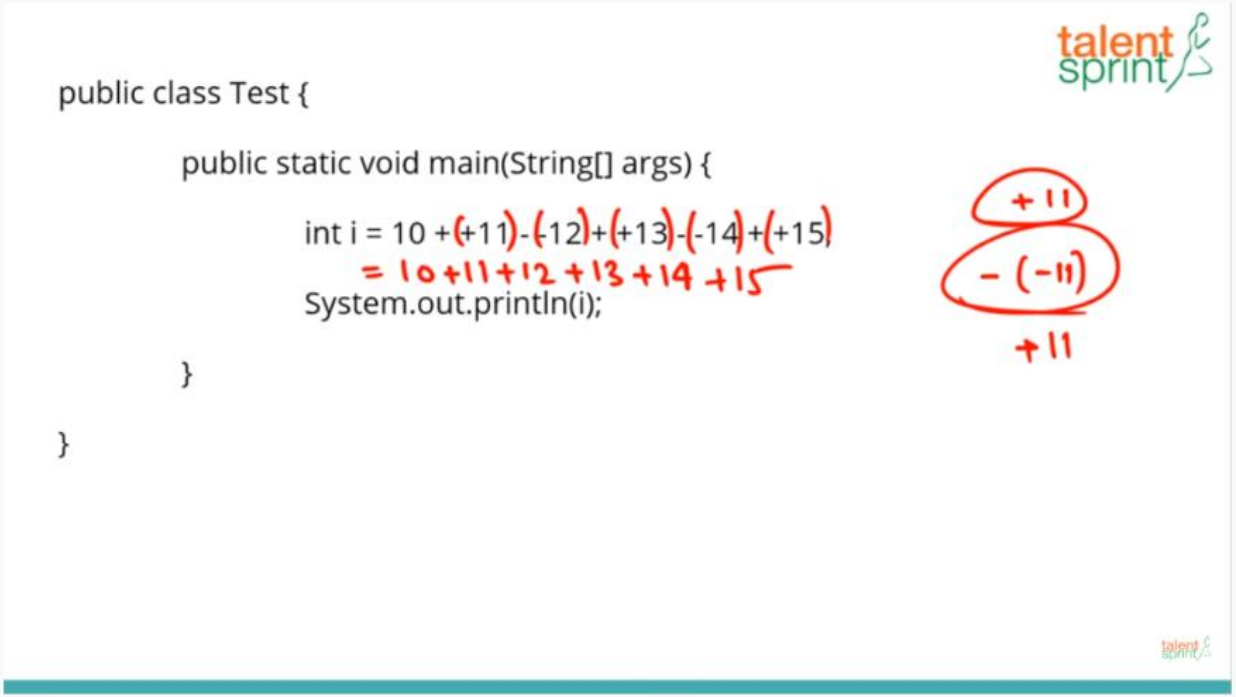
8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

Up next AUTOPLAY

Comparable vs Comparator Navin Reddy 12:58
Interview Question | Comparable vs Comparator in...
Telusko
228K views

Java Interface in Java 8 Default, Static Methods 18:20
8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview 54:17
Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

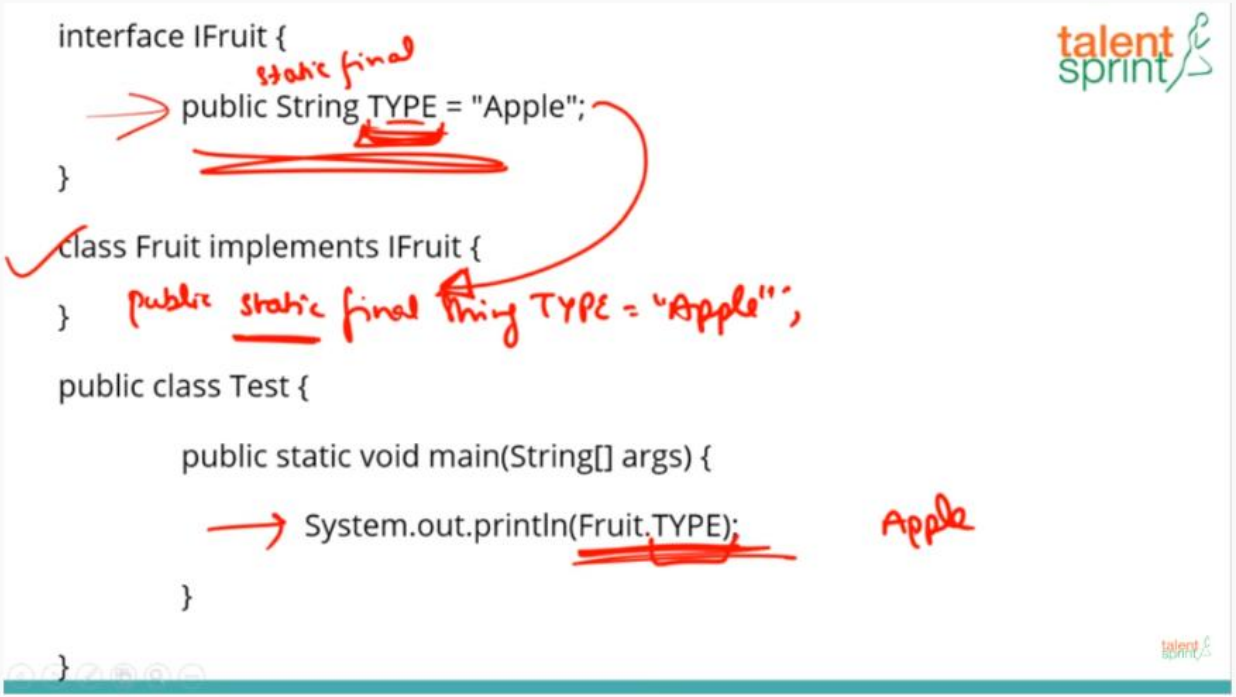
Benefits of GATE EXAM HOW TO PREPARE WITHOUT COACHING "BAAP" EXAM 21:35
Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

TOP 100 'C' MCQs 2:43:27
Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA

YouTube IN

coding interview questions



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

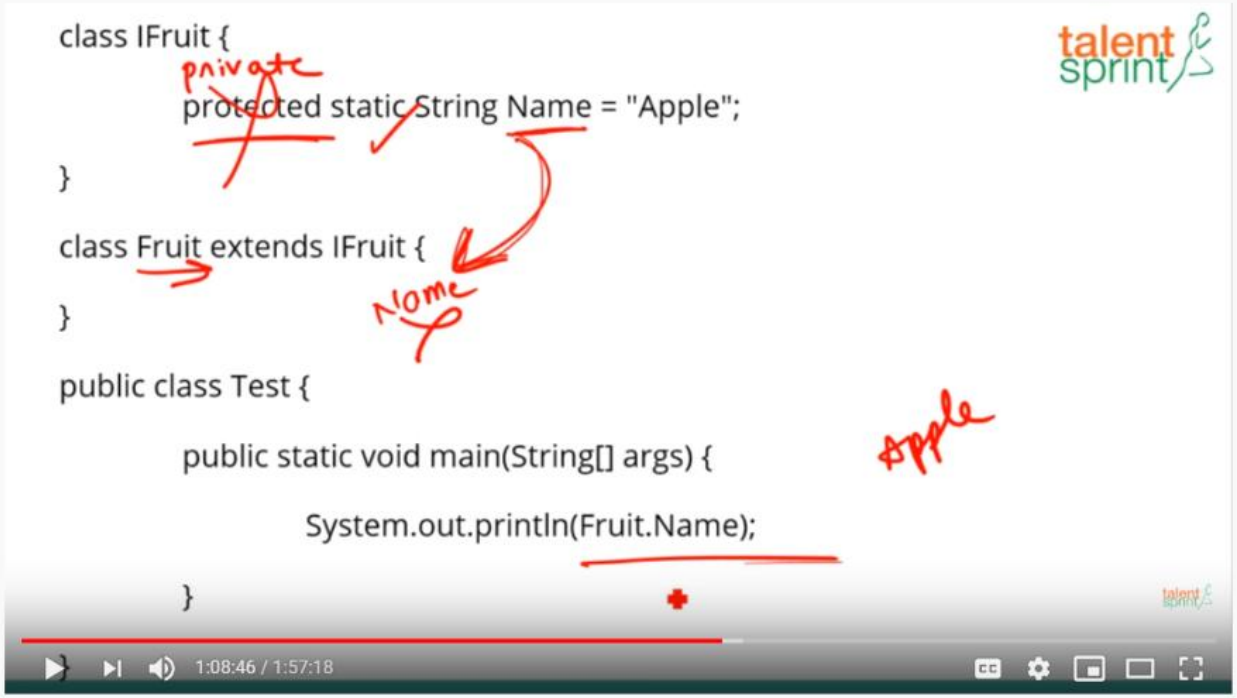
Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA

YouTube IN

coding interview questions



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA

YouTube IN

coding interview questions

```
public class Test {
        public static void main(String[] args) {
                final class Constants { public static String name = "PI"; }
                Thread thread = new Thread(new Runnable() {
                        @Override
                        public void run() {
System.out.println(Constants.name); }
                });
                thread.start();
        }
```

talent sprint

1:14:02 / 1:57:18

#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

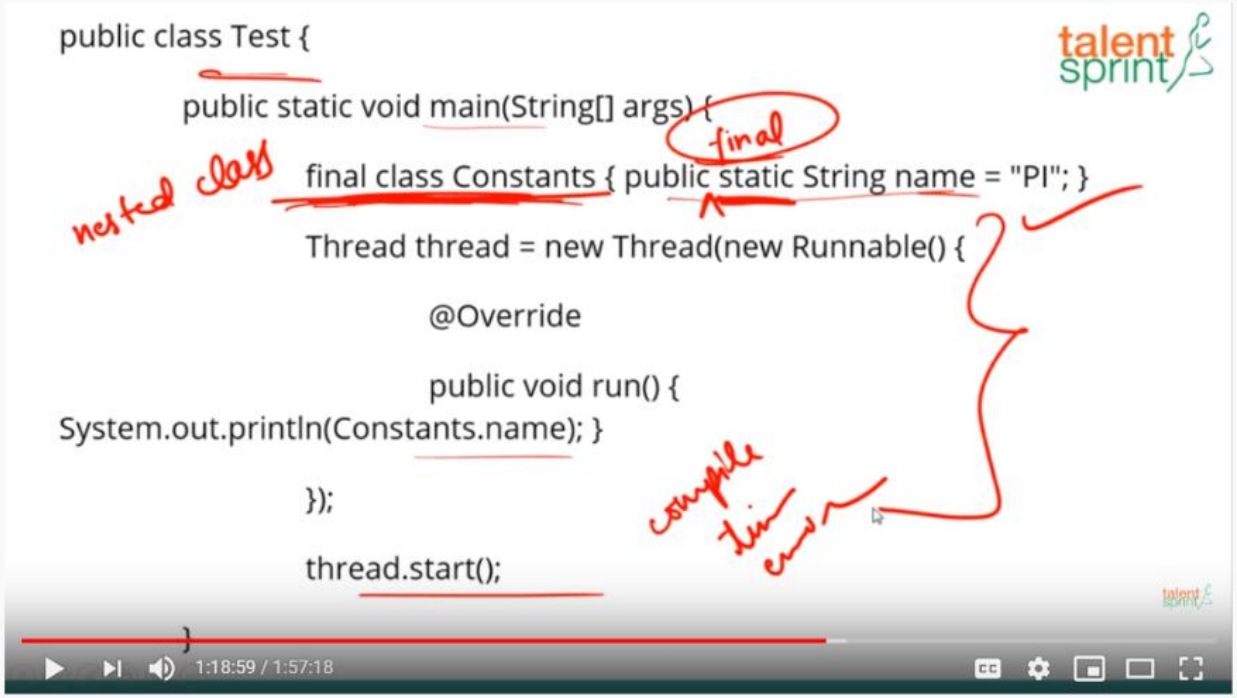
Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA

YouTube IN

coding interview questions



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

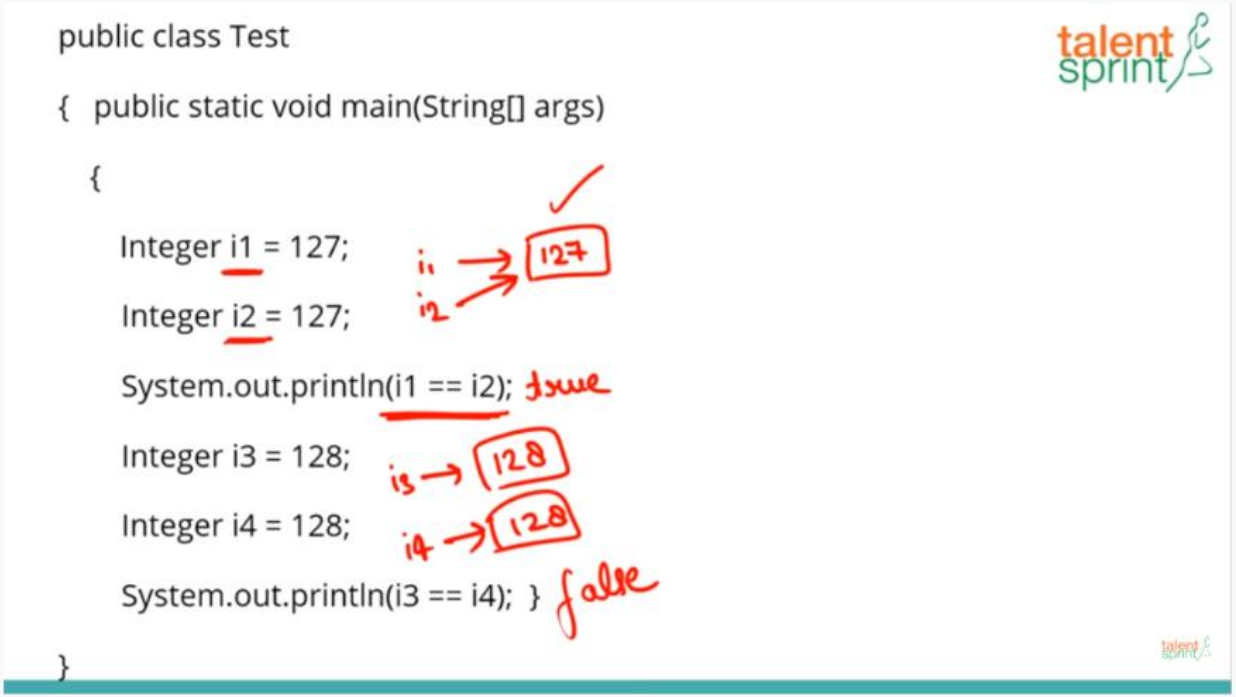
8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

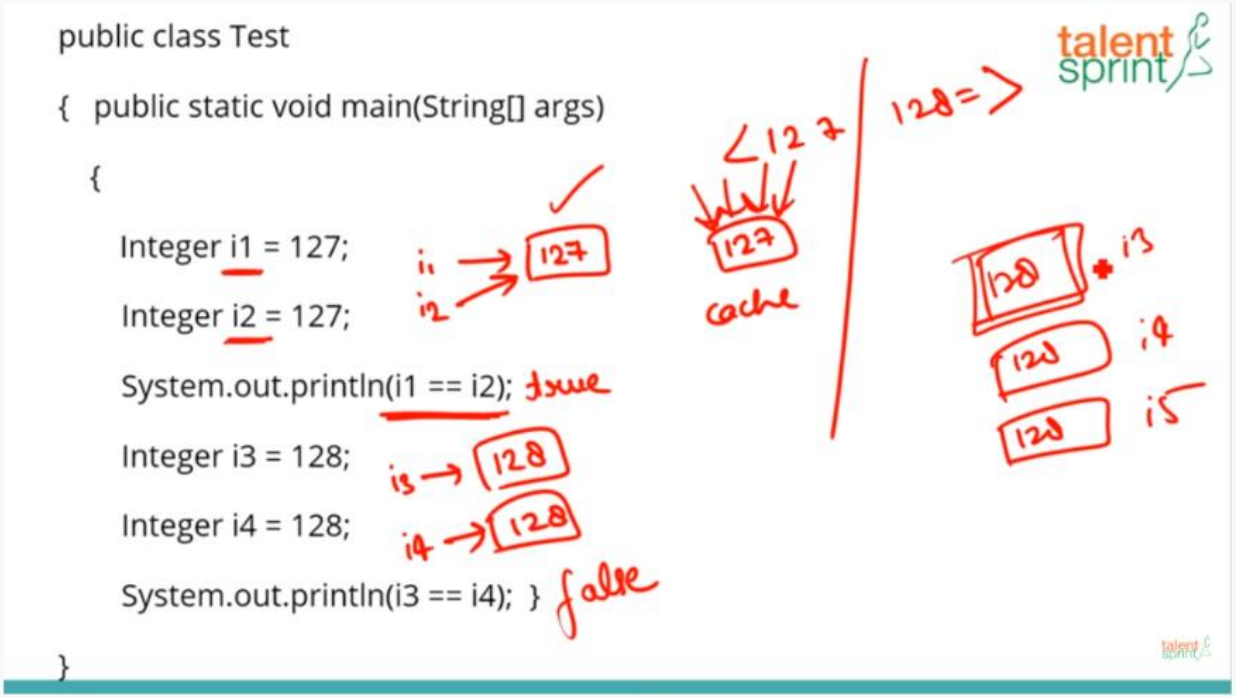
8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

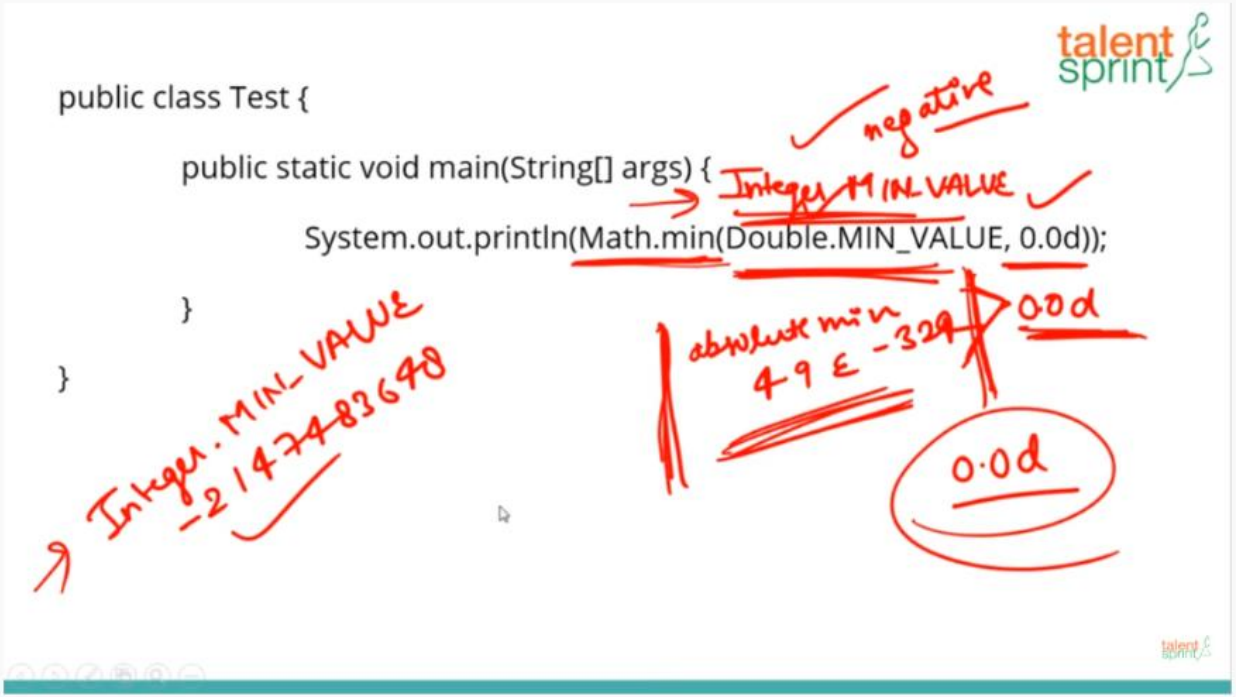
Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA

YouTube IN

coding interview questions



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

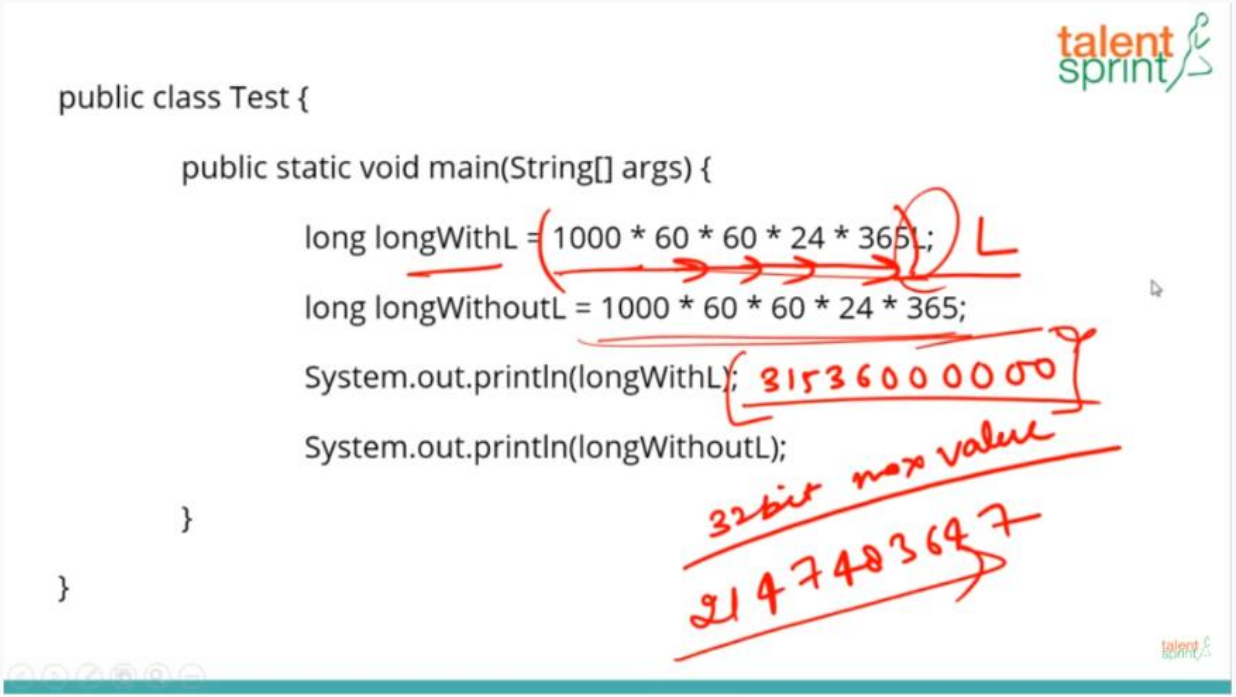
Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA

coding interview questions



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

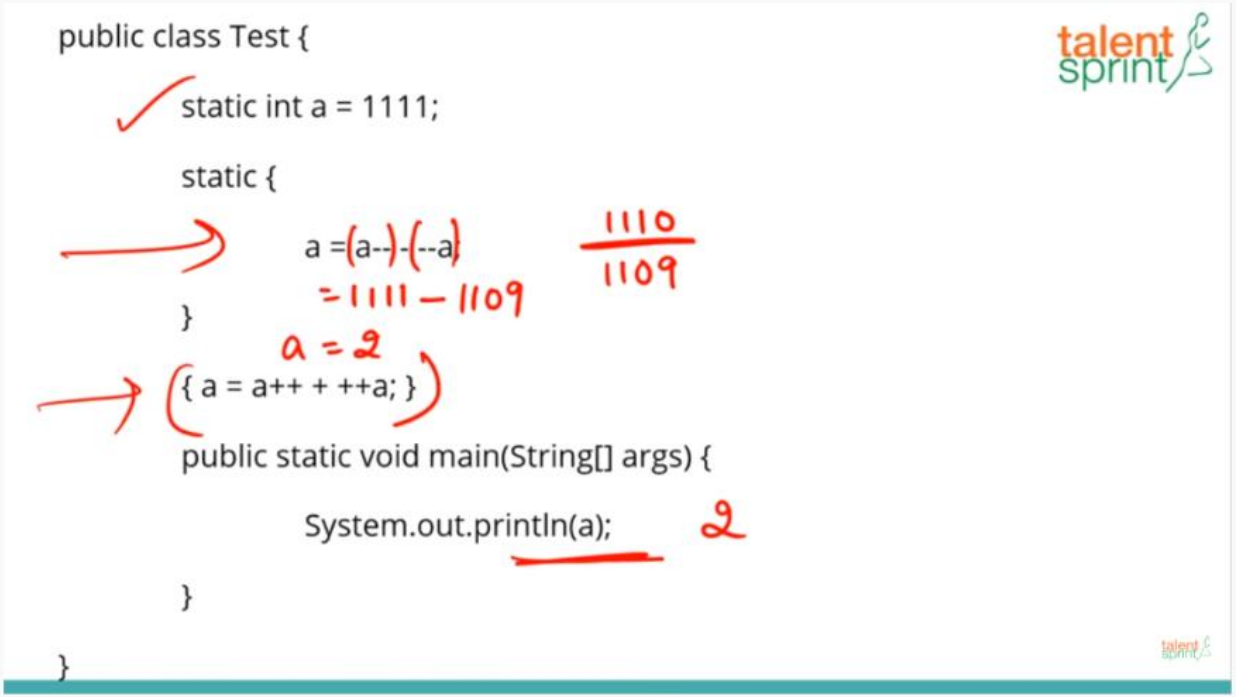
Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA

YouTube IN

coding interview questions



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

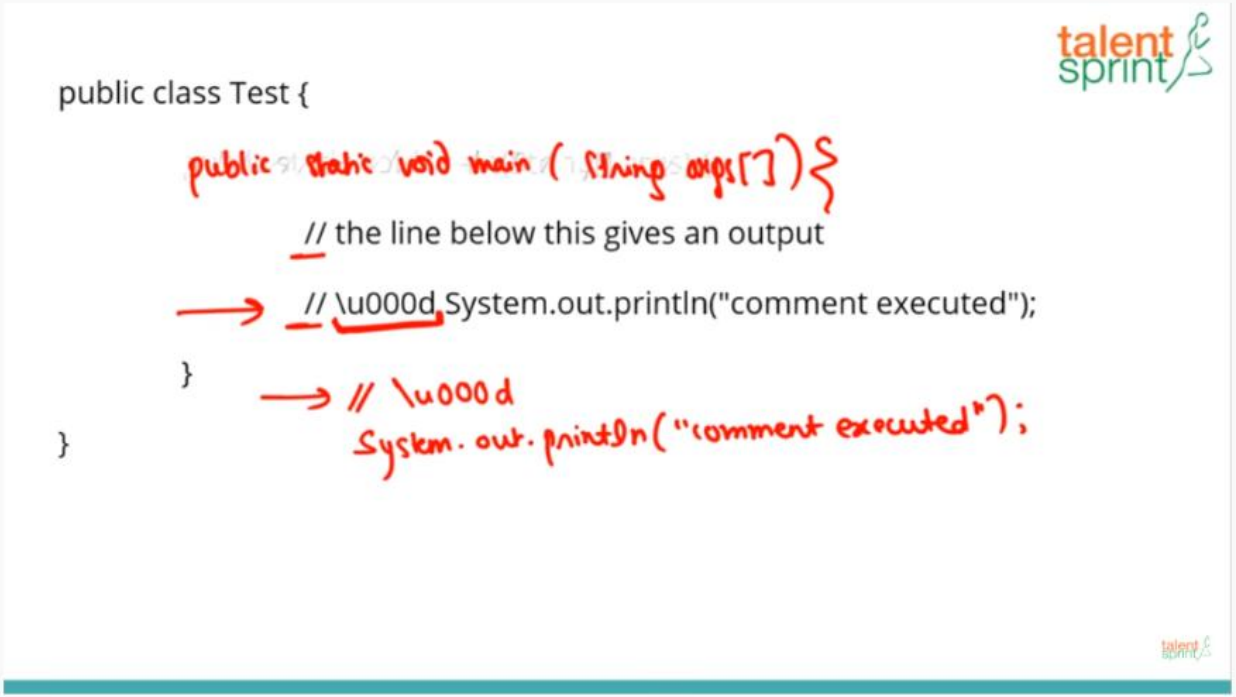
8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA



#javainterviewquestionsandanswers #javainterviewquestionsandanswersforfreshers #javainterviewquestionsforfreshers

Top 11 Tricky Java Coding Interview Questions | Java Programming | TalentSprint

188,713 views • Streamed live on Sep 25, 2017

2.2K 264 SHARE SAVE ...

Up next AUTOPLAY

Interview Question | Comparable vs Comparator in...
Telusko
228K views

8.22 Interface in Java 8 Default , Static Methods | New features
Telusko
229K views

Google Coding Interview With A Competitive Programmer
Clément Mihailescu
1.1M views

Benefits of GATE EXAM | How to Prepare WITH or WITHOUT...
Aman Dhattarwal
Recommended for you

Top 100 MCQs in C
CSE GURUS
304K views

How to Learn Anything... Fast - Josh Kaufman
The RSA

SaketSauravisaQualityAnalist

Q #19) Write a Java Program to remove all white spaces from a string without using replace().

Answer: This is another approach to removing all white spaces. Again, we have one string variable str1 with some
value. Then, we have converted that string into a character array using toCharArray().

Then, we have one StringBuffer object sb which will be used to append the value stored at chars[i] index after we have included for loop and one if condition.

If the condition is set such that then the element at i index of the character array should not be equal to space or tab. Finally, we have printed our StringBuffer object sb.

```
1  class RemoveWhiteSpaces
2  {
3      public static void main(String[] args)
4      {
5          String str1 = "Saket Saurav        is an Autom ation Engi ne      er";
6
7          char[] chars = str1.toCharArray();
8
9          StringBuffer sb = new StringBuffer();
10
11         for (int i = 0; i < chars.length; i++)
12         {
13             if( (chars[i] != ' ') && (chars[i] != '\t') )
14             {
15                 sb.append(chars[i]);
16             }
17         }
18         System.out.println(sb);           //Output :
   CoreJavajspservletsjdbcstrutshibernatespring
19     }
20 }
```

Output:

SaketSauravisanAutomationEngineer

Concrete Class

Any normal class which does not have any abstract method or a class having an implementation for all of its methods is basically a concrete class. They cannot have any unimplemented methods. A concrete class can extend its parent class, an abstract class or implement an interface if it implements all their methods. It is a complete class that can be instantiated.

Abstract Class

An abstract class is declared with an abstract keyword and have zero or more abstract methods. These classes are incomplete classes, therefore, to use an abstract class we strictly need to extend the abstract classes to a concrete class. It can have constructors and static methods as well. It can have final methods which will force the subclass to keep the body of the method unhung.

Final Class

Once a variable, method or a class is declared as final, it' s value remains the same throughout. The final keyword in a method declaration indicates that the method cannot be overridden by any subclasses i.e., a class that has been declared final cannot be subclassed. This helps a lot while creating an immutable class like the String class. A class cannot make a class immutable without making it final

In Java, There are 4 major types of classes.

what is Sealed class
What is the difference between TempData keep() and peek() function?
Why abstract class used as base class over normal class
When we have to use Interface and Abstract practically?

without making it final

0

Sep, 2019
16

In Java, There are 4 major types of classes.

1. Abstract Java Classes
   Abstract Java class can have abstract methods and non-abstract methods.
   If a class have an abstract method, this class must be abstract Java class.
   If we want to use an abstract class, it needs to be extended and its methods implemented.

2. Nested Java Classes
   Java programming language allows you to define a class within another class. We use the nested class to logically group of the class and interface in one place It can be more readable maintainable and can access all the members of the outer class(public, private, protected).

3. Final Java Classes
   When we use the final with any class, it is called a final Java class. A final class can' t be inherited. If any class is declared final then all its methods implicitly get declared as final.

4. Singletan Java class

In object-oriented programming, a singleton class is a class that can have only one object at a time. Singletons often control access to resources, such as database connections or sockets. The intention is to create a single instance of an object of classes which are expensive to create during runtime and try to reuse the same object.

For a Detailed tutorial on Java Classes, visit

Jan, 2020
8

POJO Class

POJO stands for "Plain Old Java Object". A class which contains only private variables and setter and getter methods to use those variables is called POJO class

Static Class

In Java, static is a keyword used to describe how objects are managed within the memory. A static object belongs specifically to the class, instead of instances of that class. The sole purpose of the class is to provide blueprints of its inherited classes. A static class can contain static members only. You cannot create an object for a static class.

Inner class
Inner class means the class which is a member of another class. There are four types of inner classes in java.
1) Nested Inner class
2) Method Local inner classes
3) Anonymous inner classes
4) Static nested classes

Concrete Class

Any normal class which does not have any abstract method or a class having an implementation for all of its methods is basically a concrete class. They cannot have any unimplemented methods. A concrete class can extend its parent class, an abstract class or implement an interface if it implements all their methods. It is a complete class that can be instantiated.

Sr. Developer (675)
Sr. Programmer (437)
Fresher (381)
Tech Lead (252)
DBA (240)
Team Lead (194)
Project Lead (86)
View All

MOST LIKED QUESTIONS
What are the advantages of using REST in Web API?
Why do you want to leave your current company?
What is ASP.NET Core?
Which Is Faster MVC or ASP.net ?
A class provides a default constructor for me. I write a constructor that takes a string as...
Can multiple catch blocks be executed in a C# program?
what is Sealed class
What is the difference between TempData keep() and peek() function?
Why abstract class used as base class over normal class
When we have to use Interface and Abstract practically?

Types of variable

Sep, 2019
16

In Java, There are three types of variabes, These are:

1) Local Variable
A variable, which is declared inside any methods, constructors, and blocks is known as a local variable.
Its scope is limited because it is used locally in the method. It is created after the creation of the method,
constructor, block, a local variable is not visible to class and not accessed by the class.

2) Static Variable
A static variable is a very important type of variable. We use static keyword with the variable. Hence, it' s called as a static variable.
Static Variable is declared inside the class and outside the method, constructor and a block.
Static variable belongs to the class, not an Object and it is stored in static memory. Its default value is 0 and we can call the static variable with its class name.

3) Instance Variable

An instance variable is declared inside the class but outside the method, constructor or block. It has the widest scope because it is globally visible to the whole class. It is created at the time of object creation. When instance variable is created at that time, it takes space in the heap memory and it is called through the object and the default value is 0.

For a detailed tutorial and example program on Variables in Java, Visit
https://www.c-sharpcorner.com/article/a-complete-java-classes-tutorial/

0

Programmer (1302)
Senior Management (1173)
Sr. Developer (675)
Sr. Programmer (437)
Fresher (381)
Tech Lead (252)
DBA (240)
Team Lead (194)
Project Lead (86)
View All

MOST LIKED QUESTIONS
What are the advantages of using REST in Web API?
Why do you want to leave your current company?
What is ASP.NET Core?
Which Is Faster MVC or ASP.net ?
A class provides a default constructor for me. I write a constructor that takes a string as...
Can multiple catch blocks be executed in a C# program?
what is Sealed class
What is the difference between TempData keep() and peek() function?
Why abstract class used as base class over normal class

repetition of insertion sort removes an element from the input data, inserting it into the correct position in the already-sorted list, until no input elements remain. The choice of which element to remove from the input is arbitrary, and can be made using almost any choice algorithm. You can see the code implementation below:

Code:

```
1  package com.java2novice.algos;
2
3  public class MyInsertionSort {
4
5      public static void main(String[] args) {
6
7          int[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };
8          insertionSort(input);
9      }
10
11     private static void printNumbers(int[] input) {
12
13         for (int i = 0; i < input.length; i++) {
14             System.out.print(input[i] + ", ");
15         }
16         System.out.println("\n");
17     }
18
19     public static void insertionSort(int array[]) {
20         int n = array.length;
21         for (int j = 1; j < n; j++) {
22             int key = array[j];
23             int i = j-1;
24             while ( (i > -1) && ( array [i] > key ) ) {
25                 array [i+1] = array [i];
26                 i--;
27             }
28             array[i+1] = key;
29             printNumbers(array);
30         }
31     }
32 }
```

Output:

```
2, 4, 9, 6, 23, 12, 34, 0, 1,

2, 4, 9, 6, 23, 12, 34, 0, 1,
```

Program: Write a program to check the given number is binary number or not?

Description:

The binary numeral system, or base-2 number system, represents numeric values using two symbols: 0 and 1. More specifically, the usual base-2 system is a positional notation with a radix of 2. Because of its straightforward implementation in digital electronic circuitry using logic gates, the binary system is used internally by almost all modern computers.

Code:

```
1  package com.java2novice.algos;
2
3  public class MyBinaryCheck {
4
5      public boolean isBinaryNumber(int binary){
6
7          boolean status = true;
8          while(true){
9              if(binary == 0){
10                 break;
11             } else {
12                 int tmp = binary%10;
13                 if(tmp > 1){
14                     status = false;
15                     break;
16                 }
17                 binary = binary/10;
18             }
19         }
20         return status;
21     }
22
23     public static void main(String a[]){
24         MyBinaryCheck mbc = new MyBinaryCheck();
25         System.out.println("Is 1000111 binary? :"+mbc.isBinaryNumber(1000111));
26         System.out.println("Is 10300111 binary? :"+mbc.isBinaryNumber(10300111));
27     }
28 }
```

Output:

numbers.

Description:

Write a program to find the sum of the first 1000 prime numbers.

Code:

```
1  package com.primesum;
2
3  public class Main {
4
5      public static void main(String args[]){
6
7          int number = 2;
8          int count = 0;
9          long sum = 0;
10         while(count < 1000){
11             if(isPrimeNumber(number)){
12                 sum += number;
13                 count++;
14             }
15             number++;
16         }
17         System.out.println(sum);
18     }
19
20     private static boolean isPrimeNumber(int number){
21
22         for(int i=2; i<=number/2; i++){
23             if(number % i == 0){
24                 return false;
25             }
26         }
27         return true;
28     }
29 }
```

Output:

```
3682913
```

<< Previous Program | Next Program >>

```
                int count = (line.split("\\s+")).length;
                if(count > currentMaxCount){
                    lines.clear();
                    lines.add(line);
                    currentMaxCount = count;
                } else if(count == currentMaxCount){
                    lines.add(line);
                }
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally{
            try{
                if(br != null) br.close();
            }catch(Exception ex){}
        }
    }

    public int getCurrentMaxCount() {
        return currentMaxCount;
    }

    public void setCurrentMaxCount(int currentMaxCount) {
        this.currentMaxCount = currentMaxCount;
    }

    public List<String> getLines() {
        return lines;
    }

    public void setLines(List<String> lines) {
        this.lines = lines;
    }

    public static void main(String a[]){

        MaxWordCountInLine mdc = new MaxWordCountInLine();
        mdc.readMaxLineCount("/Users/ngootooru/MyTestFile.txt");
        System.out.println("Max number of words in a line is: "+mdc.getCurrentMaxCount(
        System.out.println("Line with max word count:");
        List<String> lines = mdc.getLines();
        for(String l:lines){
            System.out.println(l);
        }
    }
}
```


Knowledge Centre

Preemptive scheduling Vs Time slicing?

Preemptive scheduling: The highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence.

Time slicing: A task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.

Famous Quotations

The pessimist complains about the wind; the optimist expects it to change; the realist adjusts the sails.

-- William Arthur Ward

```java
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class MaxWordCountInLine {

    private int currentMaxCount = 0;
    private List<String> lines = new ArrayList<String>();

    public void readMaxLineCount(String fileName){

        FileInputStream fis = null;
        DataInputStream dis = null;
        BufferedReader br = null;

        try {
            fis = new FileInputStream(fileName);
            dis = new DataInputStream(fis);
            br = new BufferedReader(new InputStreamReader(dis));
            String line = null;
            while((line = br.readLine()) != null){

                int count = (line.split("\\s+")).length;
                if(count > currentMaxCount){
                    lines.clear();
                    lines.add(line);
                    currentMaxCount = count;
                } else if(count == currentMaxCount){
                    lines.add(line);
                }
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally{
            try{
                if(br != null) br.close();
            }catch(Exception ex){}
        }
    }

    public int getCurrentMaxCount() {
```


Knowledge Centre

Preemptive scheduling Vs Time slicing?

Preemptive scheduling: The highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence.

Time slicing: A task executes for a predefined

Description:

Given array is already sorted, and it has duplicate elements. Write a program to remove duplicate elements and return new array without any duplicate elements. The array should contain only unique elements.

Code:

```
1  package com.java2novice.algos;
2
3  public class MyDuplicateElements {
4
5      public static int[] removeDuplicates(int[] input){
6
7          int j = 0;
8          int i = 1;
9          //return if the array length is less than 2
10         if(input.length < 2){
11             return input;
12         }
13         while(i < input.length){
14             if(input[i] == input[j]){
15                 i++;
16             }else{
17                 input[++j] = input[i++];
18             }
19         }
20         int[] output = new int[j+1];
21         for(int k=0; k<output.length; k++){
22             output[k] = input[k];
23         }
24
25         return output;
26     }
27
28     public static void main(String a[]){
29         int[] input1 = {2,3,6,6,8,9,10,10,10,12,12};
30         int[] output = removeDuplicates(input1);
31         for(int i:output){
32             System.out.print(i+" ");
33         }
34     }
35 }
```

Output: