**Java Notes:**

1. **OOPS concepts:** There are four object-oriented programming principles. We would be discussing each of the principle in details .
   1. **Encapsulation:**
      * 1. Encapsulation is one of fundamental principle of object-oriented programming. Encapsulation is hiding/protecting data from getting corrupted inadvertent/advertently.
        2. Encapsulation is done by making member variable as private and allowing the data accessible through method/functions this leads to encapsulation of data. Methods/function can help in checking the data for validity hence data is protected.
        3. In java encapsulation achieved by proving getter and setter method and making variables as private.
        4. With encapsulation we can make the data as read only by only providing getter methods.
        5. Here is example of encapsulation in java:
        6. “public class EncapExample{
           1. private int value;
           2. private getValue(){
           3. return value;
        7. }
        8. private int setValue(int v)
        9. if(v>0)
        10. value=v;
        11. }”
   2. **Inheritance**
   3. **Abstraction**
   4. **Polymorphism**
2. **Java basic constructs**
   1. **Data Types**
   2. **Loops**
3. **String Handling:**

Here are list of important method in the string class and their brief description. Please refer to java documentation for full list of methods.

* + - 1. [charAt](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#charAt-int-)(int index): Returns the char value at the specified index.
      2. [compareTo](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#compareTo-java.lang.String-)([String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) anotherString):Compares two strings lexicographically(alphabetically)
      3. [compareToIgnoreCase](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#compareToIgnoreCase-java.lang.String-)([String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) str)Compares two strings lexicographically, ignoring case differences.
      4. [concat](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#concat-java.lang.String-)([String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) str): Concatenates the specified string to the end of this string.
      5. [contains](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#contains-java.lang.CharSequence-)([CharSequence](https://docs.oracle.com/javase/8/docs/api/java/lang/CharSequence.html" \o "interface in java.lang) s):Returns true if and only if this string contains the specified sequence of char values.
      6. [getBytes](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#getBytes--)():Encodes this String into a sequence of bytes using the platform's default charset, storing the result into a new byte array.
      7. [indexOf](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#indexOf-int-)(int ch):Returns the index within this string of the first occurrence of the specified character.
      8. [indexOf](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#indexOf-int-int-)(int ch, int fromIndex):Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
      9. [indexOf](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#indexOf-java.lang.String-int-)([String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) str, int fromIndex):Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
      10. [isEmpty](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#isEmpty--)(): Returns true if, and only if, [length()](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#length--) is 0.
      11. [length](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#length--)():Returns the length of this string
      12. [split](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-)([String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) regex):Splits this string around matches of the given [regular expression](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#sum).
      13. [split](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-int-)([String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) regex, int limit): Splits this string around matches of the given [regular expression](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#sum).
      14. [substring](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#substring-int-)(int beginIndex):Returns a string that is a substring of this string.
      15. [substring](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#substring-int-int-)(int beginIndex, int endIndex):Returns a string that is a substring of this string.
      16. [toCharArray](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#toCharArray--)(): Converts this string to a new character array.
      17. [toLowerCase](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#toLowerCase--)(): Converts all of the characters in this String to lower case using the rules of the default locale .
      18. [toUpperCase](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#toUpperCase--)():Converts all of the characters in this String to upper case using the rules of the default locale.
      19. [trim](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#trim--)(): Returns a string whose value is this string, with any leading and trailing whitespace removed.
    1. **what is string in java ?**
       1. Strings, one of the most common objects used in Java programming, are essentially sequences of characters. As an example, the string "Scaler" contains the following characters: "S", "c", "a", "l", "e", and "r". You can either create a string by using String Literal or by using the NEW keyword. Additionally, String supports a variety of methods to operate on Strings, such as the equals method to compare two Strings, the replace method to replace String characters, the substring method to get a substring, the toUpperCase method to convert String to upper case, the split method to split a long String into multiple Strings, and so on.
       2. String is a Class in java and defined in java.lang package. It’s not a primitive data type like int and long. String class represents character Strings. String is used in almost all the Java applications and there are some interesting facts we should know about String. String in immutable and final in Java and JVM uses String Pool to store all the String objects. Some other interesting
       3. things about String is the way we can instantiate a String object using double quotes and overloading of “+” operator for concatenation.
    2. **what are different ways to create string objects ?**
       1. We can create String object using new operator like any normal java class or we can use double quotes to create a String object. There are several constructors available in String class to get String from char array, byte array, StringBuffer and StringBuilder.
       2. String str = new String("abc");
       3. String str1 = "abc";
       4. When we create a String using double quotes, JVM looks in the String pool to find if any other String is stored with the same value. If found, it just returns the reference to that String object else it creates a new String object with given value and stores it in the String pool. When we use the new operator, JVM creates the String object but don’t store it into the String Pool. We can use intern() method to store the String object into String pool or return the reference if there is already a String with equal value present in the pool.
    3. **Write a method to check if input String is Palindrome?**
       1. A String is said to be Palindrome if it’s value is same when reversed. For example “aba” is a Palindrome String. String class doesn’t provide any method to reverse the String but StringBuffer and StringBuilder class has reverse method that we can use to check if String is palindrome or not.
       2. private static boolean isPalindrome(String str) {
          1. if (str == null)
          2. return false;
          3. StringBuilder strBuilder = new StringBuilder(str);
          4. strBuilder.reverse();
          5. return strBuilder.toString().equals(str);
       3. }
       4. Sometimes interviewer asks not to use any other class to check this, in that case, we can compare characters in the String from both ends to find out if it’s palindrome or not.
       5. private static boolean isPalindromeString(String str) {

if (str == null)

return false;

int length = str.length();

System.out.println(length / 2);

for (int i = 0; i < length / 2; i++) {

if (str.charAt(i) != str.charAt(length - i - 1))

return false;

}

return true;

}

* + 1. **Write a method that will remove given character from the String?**
       1. We can use replaceAll method to replace all the occurance of a String with another String. The important point to note is that it accepts String as argument, so we will use Character class to create String and use it to replace all the characters with empty String.
       2. private static String removeChar(String str, char c) {
          1. if (str == null)
          2. return null;
          3. return str.replaceAll(Character.toString(c), "");
       3. }
    2. **How can we make String upper case or lower case?**
       1. We can use String class toUpperCase and toLowerCase methods to get the String in all upper case or lower case. These methods have a variant that accepts Locale argument and use that locale rules to convert String to upper or lower case.
    3. **What is String subSequence method?**
       1. Java 1.4 introduced CharSequence interface and String implements this interface, this is the only reason for the implementation of subSequence method in String class. Internally it invokes the String substring method. Check this post for String subSequence example.
    4. **How to compare two Strings in java program?**
       1. Java String implements Comparable interface and it has two variants of compareTo() methods. compareTo(String anotherString) method compares the String object with the String argument passed lexicographically. If String object precedes the argument passed, it returns negative integer and if String object follows the argument String passed, it returns a positive integer. It returns zero when both the String have the same value, in this case equals(String str) method will also return true. compareToIgnoreCase(String str): This method is similar to the first one, except that it ignores the case. It uses String CASE\_INSENSITIVE\_ORDER Comparator for case insensitive comparison. If the value is zero then equalsIgnoreCase(String str) will also return true. Check this post for String compareTo example.
       2. How to convert String to char and vice versa?
       3. This is a tricky question because String is a sequence of characters, so we can’t convert it to a single character. We can use use charAt method to get the character at given index or we can use toCharArray() method to convert String to character array. Check this post for sample program on converting String to character array to String.
    5. **How to convert String to byte array and vice versa?**
       1. We can use String getBytes() method to convert String to byte array and we can use String constructor new String(byte[] arr) to convert byte array to String. Check this post for String to byte array example.
    6. **Can we use String in switch case?**
       1. This is a tricky question used to check your knowledge of current Java developments. Java 7 extended the capability of switch case to use Strings also, earlier Java versions don’t support this. If you are implementing conditional flow for Strings, you can use if-else conditions and you can use switch case if you are using Java 7 or higher versions. Check this post for Java Switch Case String example.
    7. **Write a program to print all permutations of String?**
       1. This is a tricky question and we need to use recursion to find all the permutations of a String, for example “AAB” permutations will be “AAB”, “ABA” and “BAA”. We also need to use Set to make sure there are no duplicate values. Check this post for complete program to find all permutations of String.
    8. **Write a function to find out longest palindrome in a given string?**
       1. A String can contain palindrome strings in it and to find longest palindrome in given String is a programming question. Check this post for complete program to find longest palindrome in a String.
    9. **Difference between String, StringBuffer and StringBuilder?**
       1. The string is immutable and final in Java, so whenever we do String manipulation, it creates a new String. String manipulations are resource consuming, so java provides two utility classes for String manipulations - StringBuffer and StringBuilder. StringBuffer and StringBuilder are mutable classes. StringBuffer operations are thread-safe and synchronized where StringBuilder operations are not thread-safe. So in a multi-threaded environment, we should use StringBuffer but in the single-threaded environment, we should use StringBuilder. StringBuilder performance is fast than StringBuffer because of no overhead of synchronization. Check this post for extensive details about String vs StringBuffer vs StringBuilder. Read this post for benchmarking of StringBuffer vs StringBuilder.
    10. **Why String is immutable or final in Java**
        1. There are several benefits of String because it’s immutable and final.
        2. String Pool is possible because String is immutable in java.
        3. It increases security because any hacker can’t change its value and it’s used for storing sensitive information such as database username, password etc.
        4. Since String is immutable, it’s safe to use in multi-threading and we don’t need any synchronization.
        5. Strings are used in java classloader and immutability provides security that correct class is getting loaded by Classloader.
    11. **How to Split String in java?**
        1. We can use split(String regex) to split the String into String array based on the provided regular expression. Learn more at java String split.
    12. **Why Char array is preferred over String for storing password?**
        1. String is immutable in Java and stored in String pool. Once it’s created it stays in the pool until unless garbage collected, so even though we are done with password it’s available in memory for longer duration and there is no way to avoid it. It’s a security risk because anyone having access to memory dump can find the password as clear text. If we use a char array to store password, we can set it to blank once we are done with it. So we can control for how long it’s available in memory that avoids the security threat with String.
    13. **How do you check if two Strings are equal in Java?**
        1. There are two ways to check if two Strings are equal or not - using “==” operator or using equals method. When we use “==” operator, it checks for the value of String as well as the reference but in our programming, most of the time we are checking equality of String for value only. So we should use the equals method to check if two Strings are equal or not. There is another function equalsIgnoreCase that we can use to ignore case.

String s1 = "abc";

String s2 = "abc";

String s3= new String("abc");

System.out.println("s1 == s2 ? "+(s1==s2)); //true

System.out.println("s1 == s3 ? "+(s1==s3)); //false

System.out.println("s1 equals s3 ? "+(s1.equals(s3))); //true

* + 1. **What is String Pool?**
       1. As the name suggests, String Pool is a pool of Strings stored in Java heap memory. We know that String is a special class in Java and we can create String object using new operator as well as providing values in double quotes. Check this post for more details about String Pool.
    2. **What does String intern() method do?**
       1. When the intern method is invoked, if the pool already contains a string equal to this String object as determined by the equals(Object) method, then the string from the pool is returned. Otherwise, this String object is added to the pool and a reference to this String object is returned. This method always returns a String that has the same contents as this string but is guaranteed to be from a pool of unique strings.
    3. **Does String is thread-safe in Java?**
       1. Strings are immutable, so we can’t change it’s value in program. Hence it’s thread-safe and can be safely used in multi-threaded environment. Check this post for Thread Safety in Java.
    4. **Why String is popular HashMap key in Java?**
       1. Since String is immutable, its hashcode is cached at the time of creation and it doesn’t need to be calculated again. This makes it a great candidate for the key in a Map and it’s processing is fast than other HashMap key objects. This is why String is mostly used Object as HashMap keys.
    5. String Programming Questions
    6. **What is the output of below program?**

package com.journaldev.strings;

public class StringTest {

public static void main(String[] args) {

String s1 = new String("java");

String s2 = new String("PYTHON");

System.out.println(s1 = s2);

}

}

It’s a simple yet tricky program, it will print “PYTHON” because we are assigning s2 String to s1. Don’t get confused with == comparison operator.

* + 1. **What is the output of below program?**

package com.journaldev.strings;

public class Test {

public void foo(String s) {

System.out.println("String");

}

public void foo(StringBuffer sb){

System.out.println("StringBuffer");

}

public static void main(String[] args) {

new Test().foo(null);

}

}

The above program will not compile with error as “The method foo(String) is ambiguous for the type Test”. For complete clarification read Understanding the method X is ambiguous for the type Y error.

* + 1. **What is the output of below code snippet?**

String s1 = new String("abc");

String s2 = new String("abc");

System.out.println(s1 == s2);

It will print false because we are using new operator to create String, so it will be created in the heap memory and both s1, s2 will have different reference. If we create them using double quotes, then they will be part of string pool and it will print true.

* + 1. **What will be output of below code snippet?**

String s1 = "abc";

StringBuffer s2 = new StringBuffer(s1);

System.out.println(s1.equals(s2));

It will print false because s2 is not of type String. If you will look at the equals method implementation in the String class, you will find a check using instanceof operator to check if the type of passed object is String? If not, then return false.

* + 1. **What will be the output of below program?**

String s1 = "abc";

String s2 = new String("abc");

s2.intern();

System.out.println(s1 ==s2);

It’s a tricky question and output will be false. We know that intern() method will return the String object reference from the string pool, but since we didn’t assigned it back to s2, there is no change in s2 and hence both s1 and s2 are having different reference. If we change the code in line 3 to s2 = s2.intern(); then output will be true.

* + 1. **How many String objects got created in below code snippet?**

String s1 = new String("Hello");

String s2 = new String("Hello");

The answer is 3. First - line 1, “Hello” object in the string pool. Second - line 1, new String with value “Hello” in the heap memory. Third - line 2, new String with value “Hello” in the heap memory. Here “Hello” string from string pool is reused.

1. **Exception Handling**
2. **Multithreading**
3. **Collection Framework**
4. **JDBC**
5. **Servlet**
6. **JSP**