6(a)(Strings)

(1)(i) String. In Java.

→ A String in java is a sequence of characters enclosed in double quotes(" "). It is not a primitive type, but a class (java.lang.String). Strings are objects, but java lets to work with them like primitive types.

- Create a String

(i) Using String Literal
- Stored in string pool (memory - Saving Technique).

ex →

```
                        variable        object
datatype → String   Str1 = "Hello";
```

(ii) Using new keyword
- Stored in heap memory, always creates a new object.

ex →

```
String str2 = new String("Hello");
```

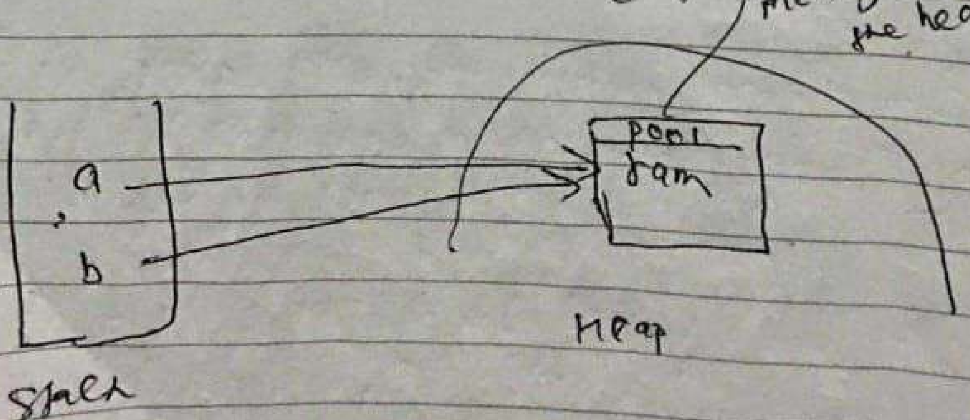
1(ii) String pool

Internal working of storing.

```
Ex- String a = "ram";
    string b = "ram";
```

(String pool is a seprate memory structure inside the heap..)

Both variable points the same object.



sout→ system.out.println ( ) ;

Date
Page No.

1(ii) String Immutability

⇒ Once a String object is created, it cannot be changed.

⇒ Any operation that seems to "Change" the String will actually Create a new String object in memory.

- Why String is immutable ?

⇒ Security , Thread - Saftey , Caching (String Pool), etc.

Example :-

```
class {
main function {
String str = "Hello" ;
str. Concat ("World") ;
Sout (str) ;      // output Hello .
```

Explanation

(i) str points to "Hello".

(ii) str. Concat ("world") creates a new String "Hello World" but doesnot assign it back to str.

(iii) str still points "Hello".

So, the original str remains unchanged.

That's immutability.

2. String Comparison

(i) = = ← [checks memory location]

⇒ It checks whether both String references point to the same memory location.

✓ Example :- 1 :

```
String a = "Test";
String b = "Test";
Sout(a = = b);  // True
```

⇒ Both a & b refer to the same object in the String Pool.

X X Example 2 :

```
String a = new String("Test");
String b = new String("Test");
Sout(a==b)      // false.
```

⇒ new keyword creates new memory objects, even if the content is same.

(ii) .equals ( ) ← [Checks Content]

→ It compares content only, not memory location — so, it returns true if characters match, ~~even if~~ regardless of how Strings are created.

✓ Example → 1)

```
String a = "Java";
String b = "Java";
Sout(a.equals(b));   //true.
```

Content is same — so returns true.

Ex-2 ->

```
String Str1 = new String ("Java");
String Str2 = new String ("Java");
Sout (Str1.equals(Str2));   //true
```

It has separate objects in heap But the content are same. So, returns true. It just checks values/obje

(3) String Concatenation Operators

-> String Concatenation means joining two or more Strings together to make one Combined String.

There are two ways to do:

(1) using + operator

(2) using the .Concat() method

Example ->

```
String a = "Hello";
String b = "World";
```

(1) //with using (+)

```
System.out.println(a+b);  //output> HelloWorld
```

(2) //with using .Concat()

```
System.out.println(a.Concat(b));  //output> Helloworld
```

(4) String MethodS

(i) length()

→ Returns the number of characters in the string.

Ex →
```
String name = "Ravi";
Sout(name.length()); //output → 4.
```

(ii) charAt(int index)

→ Returns the character at a given index (starting from 0) & (end at •length - 1).

Example →
```
String word = "Hello";
Sout(word.charAt(0)); //output → H
```

(iii) Substring(int start) and Substring(int start, int end)

→ Extracts a portion of the string from the given index range.

Example →
```
String name = "Javaprogramming";
Sout(name.substring(4)); //output → programming
Sout(name.Substring(0,4)); //output → Java
```

(iv) equalsIgnoreCase(String other)

→ Compare two strings ignoring case differences.

Ex →
```
String a = "Hello";
String b = "hello";
Sout(a.equalsIgnoreCase(b)); //output → true
```

(v) toUpperCase() and toLowerCase()

=> Returns a new String with all characters converted to uppercase or lowercase

Example => String S = "Java";
Sout(s.toUpperCase()); //output => JAVA
Sout(S.toLowerCase()); //output => java

ii) trim()

=> Removes leading and trailing spaces from a string.

Example => String text = " Hello world ";
Sout(text.trim()); //output => "Hello world";

ii) Contains(charsequence seq)

=> checks if a string contains a specific sequence of characters.

Example -> String Sentence = "Java is powerfull";
Sout(Sentence.Contains("Java")); //output => true

ii) Startwith(String prefix) and endswith(String suffix)

=> Checks if a String starts or ends with a specific word or character.

Example => String msg = "Hello World";
Sout(msg.StartsWith("Hello"); //output -> true
Sout(msg.endswith("World"); //output -> true

(ix) replace(char oldchar, char newchar) or replace(String oldStr, String newStr)

-> Replaces one character or word with another.

Example ->
```
String test = "banana";
Sout(text.replace('a', 'o'); //output->bonono
```

(x) split(String delimiter)

=> Splits a string into an array based on a pattern or character (like , or space).

Example ->
```
String fruits = "apple,banana,mango";
String[] arr = fruits.split(",");
    Sout(arr[0]);  //apple
    Sout(arr[1]);  //banana
```

(xi) toCharArray()

-> Converts a string to a character array.

Example ->
```
  String name = "Java";
char[] ch = name.toCharArray(); // convert to char Array
  for(char c : ch){
  Sout(c);        // print each character.
```

(xii) indexOf(char) and lastIndexOf(char)

-> Returns the index of the first / last occurance of a character or substring.

Example ->
```
String word = "banana";
Sout(word.indexOf('a'); //output -> 1
Sout(word.lastIndexOf('a'); //output -> 5
```

(xiii) isEmpty() -> Returns true if string is empty (" ").

5. Data Type Conversion.

Type	To String	From String
int	String.value of(int)	Integer.parseInt(str)
float	" (float)	Float.praseFloat(str)
double	" (double)	Double.parseDouble(str)
long	" (long)	Long.parseLong(str)
boolean	" (boolean)	Boolen.parseBoolean (str)
char	new string(char[])	str.tocharArray()
byte	String.value of (byte)	Byte.parseByte(str)
short	String.value of (short)	Short.parse Short (str)

(6) Java String Formatting

→ String formatting allows you to insert values(variables) into a string with specific structure and style.

1. String.format()

Ex→

```
String name = "Ravi";
int age = 21;
String result = String.format("Name : %s, Age : %d", name, age);
Sout(result); //output → Name : Ravi, Age : 21
```

(2) System.out.printf()

```
String lang = "Java";
int version = 17;
System.out.printf("Languge : %s, Version : %d, lang, version);

//output → Language : Java, Version : 17
```