CS202: IT Workshop

Java

Sorting objects and String class

Ref:

1. https://docs.oracle.com/
2. Other sources from Internet

Sorting objects

- We used ArrayList to store elements (objects)
- If we need to sort them, we can take help of the available sort () method of java (Collections.sort(), sort() method of ArrayList)
- To sort objects, there has to be some ordering among them
- We should be able to compare one object with another
- If the natural ordering is not there, we must define it

```
public int compareTo (Students st) {

        return this.age - st.age;
}
```

Return rule (ascending oder)

+ve : if curr object > arg object
-ve : if curr object < arg object
0 : if curr object = arg object

Sorting objects

- *compareTo*() is an abstract method of **Comparable** interface and the class needs to implement that

```
public class Students implements Comparable<Students> {
        . . .
        public int compareTo (Students st) {
                . . .
        }
}
```

Code (Students.java): Screen share

Sorting objects using comparator

- If we need to compare based on different fields (say, roll number, age, etc.), we need to implement interface **Comparator** and need to define method compare()

```
class StudentAgeComparator implements Comparator<Student> {

        public int compare (Student s1, Student s2) {
                return s1.age – s2.age;
        }
```

- Then we need to pass an object the comparator class

```
. . .
ArrayList<Student> sl=new ArrayList<Student>();
. . .
Collections.sort (sl, new StudentAgeComparator() );
```

Sorting objects using comparator

- We can also use them together

```
public static Comparator<Student> StudentAgeComparator = new
Comparator<Student>() {
          public int compare(Student s1, Student s2)
          {
                    return s1.age() - s2.age();
          }
};
```

Code (ComparatorDemo.java): Screen share

Questions?

String class in Java

- **Application**: Text processing, text editors, input validation, etc.
- A **sequence of characters** treated as a **single** unit
- Supports various constructors

```
String s = new String( "hello" );

String s1 = new String();

String s2 = new String( s );

char[] charArray = { ‘h', ‘e', ‘l', ‘l', ‘o', ' ‘};

String s3 = new String( charArray );

String s4 = new String( charArray, 1, 3 );
```

Created from a set of characters that is passed as an argument

Creates an object of empty string

Created from another string that is passed as an argument

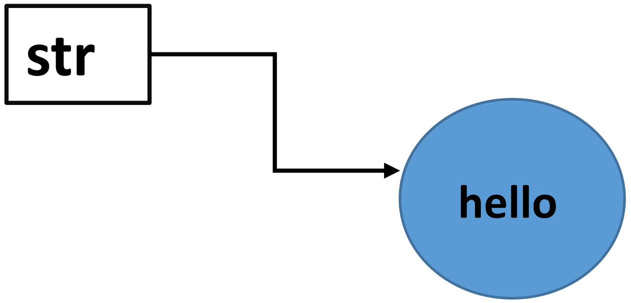
created from a char array

3 characters starting from pos 1

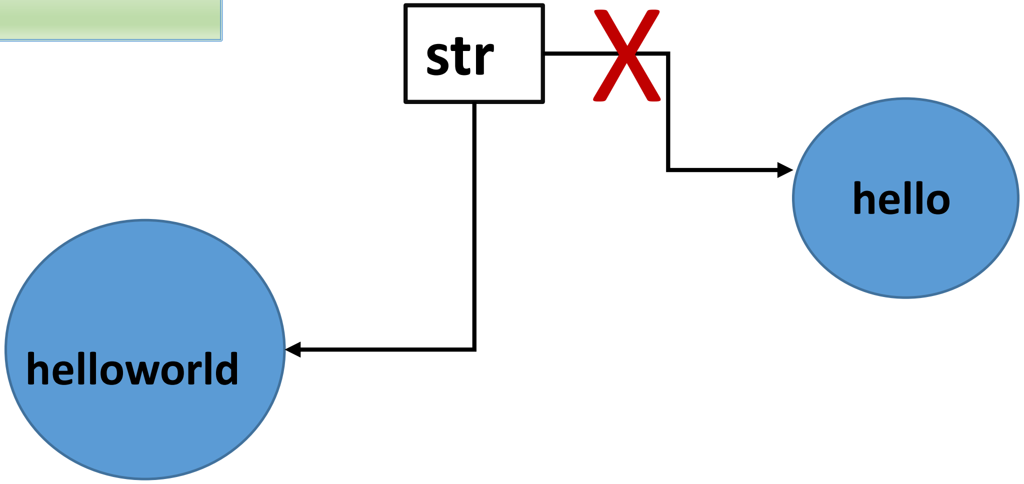
String objects are immutable

- Once created, a string cannot be changed: none of its methods changes the string

```
String str = “hello”;
```



```
str = str + “world”;
```



Supported methods in String

- Supports a number of methods: https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

```
String str = new String( "hello" );

int i = str.length();

char ch = str.charAt(2);

char ch = “iiitg”.charAt(2);

String subStr = str.substring(1, 4);

String capStr = str.toUpperCase();


str.toUpperCase().contains(“EL”);
```

```
$

$ 5

$ l

$ i

$ ell

$ HELLO


$ true
```

length() returns no. of chars

Returns char at given index

Substring from index 1 to 3

Converting to upper case

Methods can also be called in a chained way

String Builder/Buffer

- String objects are immutable; thus they are the best choice when we do not need to do modification

- If an application demands frequent **modifications**, string builder/ string buffer is the solution

```
StringBuilder buff1 = new StringBuilder();

StringBuilder buff2 = new StringBuilder( 10 );

StringBuilder buff3 = new StringBuilder( "hello" );
```

an object is created with no characters

an object is created with an initial capacity of 10 chars

an object is created with the given chars passed as argument

- Initial (default) size of String builder is 16 characters

Operations on String Builder/Buffer

- Characters can be added at last (**append**) or they can be added at a position (**insert**)

`StringBuilder buff = new StringBuilder( "hello" );`	$ **hello**
buff.***insert***(5, "world");	$ **helloworld**
buff.***append***(“ hi");	$ **helloworld hi**

- We can **delete** or **replace** characters from a string

buff.***delete***(2,5);	$ **heworld hi**
buff.***deleteCharAt***(7);	$ **heworldhi**
buff.***replace***(7, 9, "IIITG");	$ **heworldIIITG**

Tokenizing the strings

- When we need to divide a string into pieces, we can use **StringTokenizer** class of java.util package

```
String str ="Hi! I am fine!";
StringTokenizer st = new StringTokenizer (str,"!");
        while(st.hasMoreTokens()) {
                System.out.println(st.nextToken());
        }
```

```
$

$ Hi

$   I am fine
```

- We may also use ***split()*** method of String

```
String str ="Hi! I am fine!";
String [ ]tokens = str.split("!");
for (String token: tokens) {
        System.out.println(token);
        }
```

```
$

$ Hi

$   I am fine
```

Regular expressions

- We often need to validate a string entered by user

  e.g. Name should start with a capital letter,
  PIN code has to be 6 digits,
  EmailID must contain @

- We can specify a valid pattern (regular expression) using character class and quantifiers

Pattern	Meaning
[ab]	Either a or b (only one character)
[a-z]	Any character from a to z (only one character)
[^ab]	Any character except a and b (only one character)
[a-z][A-Z]	Any character from a to z followed by another character from A to Z
\d	Any digit
[ab] *	Any number of occurrences of **a or b**

→ Please refer text book (Section 16.7)/material for more notations and symbols

Regular expressions

- We can use *match*() method string to compare the pattern (expressed as regular expression)

```
firstName.matches( "[A-Z][a-zA-Z]*" );
```

Returns true if the string firstName is of the pattern:
Capital letter followed by any number of letters

```
PIN.matches( “\\d{6}" );
```

Returns true if the string PIN is of the pattern:

6 numbers of any digit

Regular expressions

- We may check the user input entered from console

```
Scanner scanner = new Scanner( System.in );

System.out.println( "Please enter first name:" );
String firstName = scanner.nextLine();

if ( firstName.matches( "[A-Z][a-zA-Z]*" ) )  {
            . . .
}
```

- We may also check the user input entered in a form (GUI interface) → we will see this later in the course!

Questions?

Topic covered and Midsem syllabus

Sl No.	Date	Topics covered
1	11/13.08.2020	Introduction about the course, Introduction to Java
2	18/20.08.2020	Basic programming constructs in Java
3	25/27.08.2020	Object oriented concepts: Class, object, constructor, Static members
4	01/03.09.2020	Inheritance, Polymorphism
5	08/10.09.2020	Abstract class, Interface, Final keyword
6	15/17.09.2020	Arrays, ArrayLists, Wrapper class, Command line argument
7	22/24.09.2020	Collections.sort, String handling

Best wishes