# Java API

Bob Singh

# String and StringBuffer

# Strings

- Strings represent a sequence of characters.
- The easiest way to represent a sequence of characters in Java is by using a character array. For example,

```
char charArray[ ] = new char [4];
charArray[0] = 'J';
charArray[1] = 'a';
charArray[2] = 'v';
charArray[3] = 'a';
```

# Strings

- Character arrays are not good enough to support the range of operations we may like to perform on strings.
- Strings are class objects and implemented using two classes, namely **`String`** and **`StringBuffer`**.
- A Java String is an instantiated object of the **`String`** class. In Java, strings are declared and created as follows:

```
String firstName = new String("J");
```

# Strings

- To find the length of the string, you can use the **`length`** method of the **`String`** class.

```
int m = firstName.length();
```

- Java strings can be concatenated using the + operator.

```
String fullName = firstName + lastName; String
city = "New" + "York";
System.out.println(city);
```

# String Arrays

- For example,
  - To create an **`itemArray`** of size 2 to hold two string constants.
  - To assign the strings to **`itemArray`**, either a **`for`** loop can be used or strings can be assigned elements by using two independent statements.

```
String itemArray[ ]=new String[3];
itemArray[0] = "Pencils";
itemArray[1] = "Paint";
```

# String Methods

- The **`String`** class defines a number of methods that allow to accomplish a variety of string manipulation tasks.
- Commonly used **`String`** methods and their descriptions:
  - **`toLowerCase()`**

    **`s2 = s1.toLowerCase();`**

    Converts the string s1 to lowercase
  - **`toUpperCase()`**

    **`s2 = s1.toUpperCase();`**

    Converts the string `s1` to uppercase

# String Methods

- **`replace()`**

  ```
  s2 = s1.replace('x', 'y');
  ```

  Replaces all instances of x with y

- **`trim()`**

  ```
  s2 = s1.trim( );
  ```

  Removes white spaces at the beginning and end of the string **`s1`**

- **`equals()`**

  ```
  s1.equals(s2);
  ```

  Returns true if **`s1`** is equal to **`s2`**.

# String Methods

- **`equalsIgnoreCase ()`**

  **`s1.equalsIgnoreCase(s2);`**

  Returns true if **`s1 = s2`**, ignoring the case of chars
- **`length ( )`**

  **`s1.length( );`**

  Returns the length of **`s1`**
- **`charAt ( )`**

  **`s1.charAt(n);`**

  Returns the character at the nth position of the string **`s1`**. The first character is at index 0.

# String Methods

- **`compareTo`**

  **`s1.compareTo(s2);`**

  Returns negative value if **`s1 < s2`**, positive if **`s1 > s2`** and 0 `if` **`s1 = s2`**

- **`concat`**

  **`s3 = s1.concat(s2);`**

  Concatenates **`s1`** and **`s2`** and resultant string is assigned to **`s3`**

- **`substring`**

  **`s1.substring(n)`**

  Returns substring starting from **`nth`** character

# String Methods

- **`valueOf`**

  ```
  s.valueOf(p);
  ```

  Creates a **`String`** object of the parameter **`p`**, which may be a simple type or an object.

- **`toString`**

  ```
  p.toString( );
  ```

  Creates a string representation of the object **`p`**

- **`indexOf`**

  ```
  s1.indexOf('x');
  ```

  Returns the position of the first occurrence of **`'x'`** in the String **`s1`**.

# String Methods (Program1.java)

- **`indexOf`**

  **`s1.indexOf('x',n)`**

  Returns the position of **`'x'`** that occurs after **`nth`** position in the string **`s1`**

- **`valueOf`**

  **`string.valueOf(Variable)`**

  Converts the parameter value to string representation.

# StringBuffer Class

- **`StringBuffer`** is a peer class of **`String`**.
- While **`String`** class creates strings of fixed lengths, **`StringBuffer`** class creates strings of flexible length that can be modified in terms of both length and content.
- We can insert characters and **`substring`** in the middle of the string or append another string to the end.

# StringBuffer Class Methods

- **`setCharAt`**

  ```
  s1.setCharAt(n, 'x');
  ```

  Modifies the nth character to **`x`**

- **`append`**

  ```
  s1.append(s2);
  ```

  Appends the string `s2` to `s1` at the end

- **`insertAt`**

  ```
  s1.insert(n, s2);
  ```

  Inserts the string `s2` at the position `n` of the string `s1`

# StringBuffer Class Methods

- **`setLength`**

  ```
  s1.setLength(n);
  ```

  Sets the length of the String **`s1`** to n. If **`n < s1.length( ),`** **`s1`** is truncated. If **`n > s1.length()`** zeros are added to s1.

# StringBuffer Class Example

```
class stringManipulation {
   public static void main(String args[]) {
   StringBuffer str = new StringBuffer("Object
   language");
System.out.println("Original String : " + str);
   System.out.println("Length of String is : "+
   str.length());
   for(int i = 0; i < str.length(); i++) {
   int p = i + 1;
   // accessing characters in a string
       System.out.println("Character at position : " +  p
   +" is : " + str.charAt(i));
} //Contd.
```

# StringBuffer Class Example(Program2.java)

```
 String aString = new String(str.toString());
int pos = aString.indexOf("language");
// inserting a string in the middle
str.insert(pos, "Oriented");
System.out.println("Modified string : " + str);
// Modifying character at position 6
str.setCharAt(6, '`');
System.out.println("String now : " + str);
// Appending a string at the end
str.append("improves security : ");
System.out.println("Append string : "+str);
} } //class
```

# File IO

# java.io API

- Text Streams and Buffers (Program3.java)
- Binary Streams and Buffers (Program4.java)
- Filters (Program5.java)
- Object Files (Program6.java)

# Reflections

# Reflection (Program7.java)

- The reflection API reflects the classes, interfaces, and objects in the current JVM.
- With the reflection API, you can
  - Determine the class of an object.
  - Class’s modifiers, fields, methods, constructors, and superclasses.
  - Create an instance of a class whose name is not known until runtime.
  - What constants and method declarations belong to an interface.
  - Get and set the value of an object's field, even if the field name is unknown to your program until runtime.
  - Invoke a method on an object, even if the method is not known until runtime.