Introduction to Computer Science I

String and Random Java Classes

Janyl Jumadinova
September 26-28, 2016
Divide and Conquer

- Most programs are complex and involved.
- The best way to develop and maintain a large system is to break it down and build it from smaller pieces or modules.
- The divide and conquer technique of solving problems lends itself nicely to modular program development.
Modularity in Java comes in three varieties: methods, classes and packages:

- **Methods** modularize a program by separating tasks into self-contained units.
Modularity in Java comes in three varieties: methods, classes and packages:

- **Methods** modularize a program by separating tasks into self-contained units.
- **Classes** group together related methods.
Modularity in Java comes in three varieties: methods, classes and packages:

- **Methods** modularize a program by separating tasks into self-contained units.
- **Classes** group together related methods.
- **Packages** group together related classes.
Modularity in Java comes in three varieties: methods, classes and packages:

- **Methods** modularize a program by separating tasks into self-contained units.
- **Classes** group together related methods.
- **Packages** group together related classes.
- Programs are pieced together from user-written methods and classes and pre-written methods and classes from the Java Application Programming Interface, or Java API.
The Java API (Application Programming Interface) contains many separate packages that can be used to make writing complex programs easier.

- Each package focuses on a specific set of tasks and provides pre-written methods:
  - `java.applet` - allows creation of web applets
  - `java.net` - Java networking package
  - `java.text` - Manipulate numbers, dates, etc.
  - `java.util` - Utilities (Scanner, Random, etc)
  - `javax.swing` - Swing GUI Components (JOptionPane)
  - `javax.swing.event` - Enable event handling
String class

- String str = ‘‘abc’’; is equivalent to:
  String str = new String(‘‘abc’’);
- The + operator joins two strings together.
String class

- String str = ‘‘abc’’; is equivalent to:
  String str = new String(‘‘abc’’);
- The + operator joins two strings together.
- String class is a part of the java.lang package.
- The classes of java.lang package are automatically available for use, no need to import.
String methods

**charAt()**

The **charAt()** function returns the character located at the specified index.

```java
String str = "studytonight";
System.out.println(str.charAt(2));
```

**Output:** u
String methods

**charAt()**

The `charAt()` function returns the character located at the specified index.

```java
String str = "studytonight";
System.out.println(str.charAt(2));
```

Output: u

**length()**

The `length()` function returns the number of characters in a String.

```java
String str = "Count me";
System.out.println(str.length());
```

Output: 8
String methods

**replace()**

*replace()* method replaces occurrences of character with a specified new character.

```java
String str = "Change me";
System.out.println(str.replace('m','M'));
```

*Output: Change Me*
String methods

replace()

replace() method replaces occurrences of character with a specified new character.

```java
String str = "Change me";
System.out.println(str.replace('m','M'));
```

Output: Change Me

String str1 = new String(‘‘This is really fun!!’’);
String str2 = str1.replace( ’i’, ’u’ );
String methods

replace()

replace() method replaces occurrences of character with a specified new character.

```java
String str = "Change me";
System.out.println(str.replace('m','M'));
```

Output: Change Me

String str1 = new String("This is really fun!!");
String str2 = str1.replace( 'i', 'u' );

str2 returns "Thus us really fun"
String methods

**substring()**

*substring()* method returns a part of the string. *substring()* method has two forms,

```java
public String substring(int begin);
public String substring(int begin, int end);
```

The first argument represents the starting point of the substring. If the *substring()* method is called with only one argument, the substring returned, will contain characters from specified starting point to the end of original string.

But, if the call to *substring()* method has two arguments, the second argument specify the end point of substring.

```java
String str = "0123456789";
System.out.println(str.substring(4));
```

Output: 456789
String methods

- **equals()**: This method returns true if the String are equal; false otherwise.
String methods

- `equals()`: This method returns true if the String are equal; false otherwise.

```java
String str1 = new String("This is really fun!!");
String str2 = new String("This is really fun!!");
boolean value = str1.equals(str2);
```
String methods

- `equals()`: This method returns true if the String are equal; false otherwise.

```java
String str1 = new String('"This is really fun!!"');
String str2 = new String('"This is really fun!!"');
boolean value = str1.equals( str2 );
```

- `value` returns `value = true`
The Random class is part of the java.util package.

- It provides methods that generate pseudorandom numbers.
Random Class

- The Random class is part of the java.util package
- It provides methods that generate pseudorandom numbers
- Before you can use its methods, you must create an instance of the Random class

```java
Random r = new Random();
```
Random Class

```java
float nextFloat()
float nextDouble()
- Returns a random number between 0.0 inclusive and 1.0 exclusive.
```
Random Class

```java
float nextFloat()
float nextDouble()
- Returns a random number between 0.0 inclusive and 1.0 exclusive.
Random r = new Random();
float f;
f = r.nextFloat();
```
Random Class

int nextInt( ) - Returns a random number that ranges over all possible int values positive and negative.
intnextInt( ) - Returns a random number that ranges over all possible int values positive and negative.
Random r = new Random( );
int i;
i = r.nextInt( );
Random Class

```java
int nextInt( ) - Returns a random number that ranges over all possible int values positive and negative.
Random r = new Random( );
int i;
i = r.nextInt( );
```

```java
int nextInt( int num ) - Returns a random number between 0 (inclusive) and num (exclusive).
```

Random Class

```java
int nextInt( ) - Returns a random number that ranges over all possible int values positive and negative.
Random r = new Random( );
int i;
i = r.nextInt( );
```

```java
int nextInt( int num ) - Returns a random number between 0 (inclusive) and num (exclusive).
Random r = new Random( );
int i;
i = r.nextInt(5 );
```