Computational Expression

String Class

Janyl Jumadinova

23 September, 2019
Data Types: Review

- Java has two categories of data:
  - primitive data (e.g., number, character)
  - object data (programmer created types)
- Primitive data are only single values; they have no special capabilities.
What is an Object?

- Everything is an object.
- Everything in the physical world can be described in terms of the set of characteristics and the set of behaviors it possesses.
What is an Object?

- Everything is an object.
- Everything in the physical world can be described in terms of the set of characteristics and the set of behaviors it possesses.

<table>
<thead>
<tr>
<th>Characteristics (Attributes)</th>
<th>Behaviors (Actions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Cave man</td>
<td>Calculating</td>
</tr>
<tr>
<td>Color: White</td>
<td>Standing</td>
</tr>
<tr>
<td>Hair: Orange</td>
<td></td>
</tr>
</tbody>
</table>
What is an Object?

- An object is any item that can be described by listing a set of attributes or characteristics, and a set of behaviors or actions.
- The behaviors typically involve, in some fashion, the attributes, either by using their values, examining their values or modifying their values.
- Attributes/characteristics are also called the data members of the object.
- Behaviors are also called the member functions or methods of the object.
Messages

- To communicate with an object, you send messages or requests to the object.
- For example, a television is an object.
Messages

- To communicate with an object, you send messages or requests to the object.
- For example, a television is an object.
- What are the characteristics and behaviors of a television?
- How do we send messages?
Messages

- To communicate with an object, you send messages or requests to the object.
- For example, a television is an object.
- What are the characteristics and behaviors of a television?
- How do we send messages?
- A remote control is an object that packages up requests and sends them to the television object.
Encapsulation
Grouping the characteristics and behaviors together in one entity (a class or object).
OO-related Concepts

- **Encapsulation**
  Grouping the characteristics and behaviors together in one entity (a class or object).

- **Information Hiding**
  Protecting the data members/characteristics by placing them in a private section of the class/object.
OO-related Concepts

- **Encapsulation**
  Grouping the characteristics and behaviors together in one entity (a class or object).

- **Information Hiding**
  Protecting the data members/characteristics by placing them in a private section of the class/object.

- **Public Interface**
  The public behaviors that are allowed to be used by other objects to interact with the object itself.
A class is

- A class is a template for an object.
- A class specifies the set of characteristics and the set of behaviors that an object will have.
A class is

- A class is a template for an object.
- A class specifies the set of characteristics and the set of behaviors that an object will have.
- Any object of a particular class will have the same set of characteristics and behaviors.
A class is

- A class is a template for an object.
- A class specifies the set of characteristics and the set of behaviors that an object will have.
- Any object of a particular class will have the same set of characteristics and behaviors.
- However, the values of the characteristics may differ from object to object within a particular class.
A class is a user-defined data type that will complement the native data types that are already built in to Java.

An object is an instantiation (specific instance) of a class.
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.
Java API Packages

- 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.lang` - Fundamentals (Object, String, etc)
  - `java.util` - Utilities (Scanner, Random, etc)
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:
  ```java
  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

```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.

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()` 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

```java
System.out.println(str.substring(4,7));
```

Output : 456
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`