1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Lecture 32: Advanced string and character operations

Tokenising strings

- It is often useful to break up long strings into smaller pieces, or *tokens*, for example to extract individual words from a sentence
  - This process is called ***tokenisation***
- Method **`split()`** of class `String` breaks a string into its components (tokens)
  - Tokens are separated from one another by delimiters, typically white-space characters such as *space*, *tab*, *newline*, and *carriage return*
  - Other characters can also be used as delimiters to separate tokens
  - The arguments for `split()` include delimiting *regular expression* and the optional maximum limit for the number of tokens

Tokenising example 1

```
1  import java.util.Scanner;
2  public class TokenizingExample1 {
3    public static void main(String[] args){
4      Scanner scanner = new Scanner(System.in);
5      System.out.println("Enter a sentence and press Enter");
6      String sentence = scanner.nextLine();
7      String[] tokens = sentence.split(" ");
8      System.out.printf("Number of tokens: %d%n", tokens.length);
9      System.out.println("The tokens are:");
10     for (String token : tokens) {
11       System.out.println(token);
12     }
13   }
14 }
```

Delimiter is space “ ”

```
$ java TokenizingExample1
Enter a sentence and press Enter:
I like Java
Number of elements: 3
The tokens are:
I
like
Java
$
```

Tokenising example 2

```
1  import java.util.Scanner;
2  public class TokenizingExample2 {
3    public static void main(String[] args){
4      Scanner scanner = new Scanner(System.in);
5      System.out.println("Enter your email address:");
6      String sentence = scanner.nextLine();
7      String[] tokens = sentence.split("@",2);
8      System.out.printf("Your user name is: %s%n", tokens[0]);
9      System.out.printf("Your URL is: %s%n", tokens[1]);
10   }
11 }
```

Delimiter is “@”, maximum of 2 tokens are extracted

```
$ java TokenizingExample2
Enter your email address:
jamesbond007@abdn.ac.uk
Your user name is: jamesbond007
Your URL is: abdn.ac.uk
$
```

Concatenating strings

- Method **`concat()`** of `String` can be used to concatenate two `String` objects into a new `String` object containing characters from both strings
  - Syntax: `s1.concat(s2)` will concatenate `String` objects `s1` and `s2` so that `s2` appears after `s1`
- In Java, addition operator + used on `String` objects is defined to perform concatenation
  - Assuming that `s1`, `s2`, and `s3 are string objects`:

    `s1+s2` equals to `s1.concat(s2)`

    `s1+s2+s3` equals to `s1.concat(s2).concat(s3)`

Concatenation example

```
1  public class TokenizingExample2 {
2    public static void main(String[] args){
3      String s1 = "Good ";
4      String s2 = "morning ";
5      String s3 = "world!";
6      System.out.println(s1+s2+s3);
7      System.out.println(s1.concat(s2).concat(s3));
8      int age = 18;
9      System.out.println("Michael is " + age + " years old");
10   }
11 }
```

```
$ java TokenizingExample2
Good morning world!
Good morning world!
Michael is 18 years old
$
```

Note that if the data type is not a `String` object, Java converts it automatically to a string representation when operator + is used, but it does not work with `concat()`!

Using StringBuilder for modifiable strings

- In programs that frequently perform string concatenations or other string modifications, it is often more efficient to use class **`StringBuilder`** instead of class `String`
  - `StringBuilder` is a "modifiable" version of `String`: it provides methods such as **`append()`**, **`insert()`**, and **`delete()`** to modify the contents of the string it contains
  - When a `StringBuilder` object is modified, it does not return a new `StringBuffer` object but changes the contents of the original one

StringBuilder constructors

- `StringBuilder` class provides several different constructors:
  - **`StringBuilder()`**: constructs a string builder with no characters and the initial capacity of 16 characters
  - **`StringBuilder(`**`CharSequence seq`**`)`**: constructs a string builder that contains the same characters as the `CharSequence` object `seq`
  - **`StringBuilder(`**`int capacity`**`)`**: constructs a string builder with no characters and the initial capacity specified by the `capacity` argument
  - **`StringBuilder(`**`String str`**`)`**: constructs a string builder initialised to the contents of the `String` argument `str`

StringBuilder methods

- Some of the most essential `StringBuilder` classes are the following:
  - **`length()`**, **`setLength(`**`int length`**`)`**: returns the length (character count), and sets the length, respectively
  - **`capacity()`** and **`ensureCapacity()`**: returns the current capacity, and increases capacity if below the specified minimum capacity, respectively
  - **`charAt()`**, **`setCharAt()`**, **`getChars()`**: getting and setting characters at specified positions
  - **`append()`**, **`insert()`**, **`delete()`**, **`deleteCharAt()`**: modifying the string builder by appending, inserting, and deleting content; there are several overloaded versions of these methods to support different data types

String builder example (1)

```
1  public class StringBuilderExample1 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s | length = %d | capacity = %d%n",
5          sb.toString(), sb.length(), sb.capacity());
6      sb.ensureCapacity(75);
7      System.out.printf("New capacity = %d%n", sb.capacity());
8      sb.setLength(10);
9      System.out.printf("New buffer = %s | new length = %d%n",
10         sb.toString(), sb.length());
11   }
12 }
```

String builder example (2)

```
1  public class StringBuilderExample1 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s | length = %d | capacity = %d%n",
5          sb.toString(), sb.length(), sb.capacity());
6      sb.ensureCapacity(75);
7      System.out.printf("New capacity = %d%n", sb.capacity());
8      sb.setLength(10);
9      System.out.printf("New buffer = %s | new length = %d%n",
10         sb.toString(), sb.length());
11   }
12 }
```

```
$ java StringBuilderExample1
Buffer = Good morning world! | length = 19 | capacity = 35
```

String builder example (3)

```
1  public class StringBuilderExample1 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s | length = %d | capacity = %d%n",
5          sb.toString(), sb.length(), sb.capacity());
6      sb.ensureCapacity(75);
7      System.out.printf("New capacity = %d%n", sb.capacity());
8      sb.setLength(10);
9      System.out.printf("New buffer = %s | new length = %d%n",
10         sb.toString(), sb.length());
11   }
12 }
```

```
$ java StringBuilderExample1
Buffer = Good morning world! | length = 19 | capacity = 35
New capacity = 75
```

String builder example (4)

```
1  public class StringBuilderExample1 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s | length = %d | capacity = %d%n",
5          sb.toString(), sb.length(), sb.capacity());
6      sb.ensureCapacity(75);
7      System.out.printf("New capacity = %d%n", sb.capacity());
8      sb.setLength(10);
9      System.out.printf("New buffer = %s | new length = %d%n",
10         sb.toString(), sb.length());
11   }
12 }
```

```
$ java StringBuilderExample1
Buffer = Good morning world! | length = 19 | capacity = 35
New capacity = 75
New buffer = Good morni | new length = 10
$
```

String builder example 2 (1)

```
1  public class StringBuilderExample2 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s%n", sb.toString());
5      char[] charArray = new char[7];
6      sb.getChars(5, 12, charArray, 0);
7      System.out.printf("Char array = ");
8      for (char character : charArray) {System.out.print(character);}
9      sb.setCharAt(5,'M');
10     sb.setCharAt(13,'W');
11     System.out.printf("%nNew Buffer = %s%n", sb.toString());
12     sb.deleteCharAt(sb.length()-1);
13     sb.append(" Again!");
14     System.out.printf("New Buffer = %s%n", sb.toString());
15   }
16 }
```

String builder example 2 (2)

```
1  public class StringBuilderExample2 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s%n", sb.toString());
5      char[] charArray = new char[7];
6      sb.getChars(5, 12, charArray, 0);
7      System.out.printf("Char array = ");
8      for (char character : charArray) {System.out.print(character);}
9      sb.setCharAt(5,'M');
10     sb.setCharAt(13,'W');
11     System.out.printf("%nNew Buffer = %s%n", sb.toString());
12     sb.deleteCharAt(sb.length()-1);
13     sb.append(" Again!");
14     System.out.printf("New Buffer = %s%n", sb.toString());
15   }
16 }
```

```
$ java StringBuilderExample2
Buffer = Good morning world!
```

String builder example 2 (3)

```
1  public class StringBuilderExample2 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s%n", sb.toString());
5      char[] charArray = new char[7];
6      sb.getChars(5, 12, charArray, 0);
7      System.out.printf("Char array = ");
8      for (char character : charArray) {System.out.print(character);}
9      sb.setCharAt(5,'M');
10     sb.setCharAt(13,'W');
11     System.out.printf("%nNew Buffer = %s%n", sb.toString());
12     sb.deleteCharAt(sb.length()-1);
13     sb.append(" Again!");
14     System.out.printf("New Buffer = %s%n", sb.toString());
15   }
16 }
```

```
$ java StringBuilderExample2
Buffer = Good morning world!
Char array = morning
```

String builder example 2 (4)

```
1  public class StringBuilderExample2 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s%n", sb.toString());
5      char[] charArray = new char[7];
6      sb.getChars(5, 12, charArray, 0);
7      System.out.printf("Char array = ");
8      for (char character : charArray) {System.out.print(character);}
9      sb.setCharAt(5,'M');
10     sb.setCharAt(13,'W');
11     System.out.printf("%nNew Buffer = %s%n", sb.toString());
12     sb.deleteCharAt(sb.length()-1);
13     sb.append(" Again!");
14     System.out.printf("New Buffer = %s%n", sb.toString());
15   }
16 }
```

```
$ java StringBuilderExample2
Buffer = Good morning world!
Char array = morning
New Buffer = Good Morning World!
```

String builder example 2 (5)

```
1  public class StringBuilderExample2 {
2    public static void main(String[] args){
3      StringBuilder sb = new StringBuilder("Good morning world!");
4      System.out.printf("Buffer = %s%n", sb.toString());
5      char[] charArray = new char[7];
6      sb.getChars(5, 12, charArray, 0);
7      System.out.printf("Char array = ");
8      for (char character : charArray) {System.out.print(character);}
9      sb.setCharAt(5,'M');
10     sb.setCharAt(13,'W');
11     System.out.printf("%nNew Buffer = %s%n", sb.toString());
12     sb.deleteCharAt(sb.length()-1);
13     sb.append(" Again!");
14     System.out.printf("New Buffer = %s%n", sb.toString());
15   }
16 }
```

```
$ java StringBuilderExample2
Buffer = Good morning world!
Char array = morning
New Buffer = Good Morning World!
New Buffer = Good Morning World Again!
```

Class StringBuffer

- String builders created using `StringBuilder` class *are not thread safe*: if multiple threads require access to the same dynamic (i.e., modifiable) string content, use class **`StringBuffer`** instead of `StringBuilder`
  - Both classes `StringBuilder` and `StringBuffer` provide identical capabilities, but only `StringBuffer` is thread safe (i.e., synchronized)
  - If you do *not* need to access the same string builder from multiple threads, `StringBuilder` class works faster and more efficiently than `StringBuffer`

Wrapper classes

- In some situations, you need to treat primitive type values as objects (i.e., reference type values)
  - Java provides wrapper classes **`Boolean`**, **`Character`**, **`Double`**, **`Float`**, **`Byte`**, **`Short`**, and **`Integer`** for primitive types `boolean`, `char`, `double`, `float`, `byte`, `short` and `int`, respectively
  - The recommended way to covert primitive types to wrapper class objects is to use each class's static method `valueOf()`, for example:

```
int iPrim = 1; Integer i = Integer.valueOf(iPrim);
```

  - Wrapper classes can also be initialised directly by using literals (*autoboxing*):

```
Character c = 'A'; Integer i = 5;
```

Methods of class Character

- Methods of class `Character` can be useful for testing and manipulating individual character values
  - Each method takes at least a character as input argument
  - Examples of methods for *testing* characters include: **`isDefined()`**, **`isDigit()`**, **`isJavaIdentifierStart()`**, **`isJavaIdentifierPart()`**, **`isLetter()`**, **`isLetterOrDigit()`**, **`isLowerCase()`**, and **`isUpperCase()`**
  - Example of methods for manipulating characters include: **`toUpperCase()`**, returning an uppercase version of the character, and **`toLowerCase()`**, returning a lowercase version of the character

Character example (1)

```
1  public class CharacterExample {
2    public static void main(String[] args){
3      char c = 'a';
4      Character c1 = 'A';
5      Character c2 = Character.valueOf(c);
6      System.out.printf("c1 = %c | c2 = %s%n", c1, c2.toString());
7      System.out.printf("c1 and c2 are equal? %b%n", c1.equals(c2));
8      System.out.printf("c1 and c2 are equal when case ignored? %b%n",
9                        c1.toString().equalsIgnoreCase(c2.toString()));
10     System.out.printf("'%c' is digit? %b%n", c1, Character.isDigit(c1));
11     System.out.printf("'%c' is letter? %b%n", c1, Character.isLetter(c1));
12     System.out.printf("'%c' is uppercase? %b%n", c1, Character.isUpperCase(c1));
13     System.out.printf("'%c' is digit? %b%n", c2, Character.isUpperCase(c2));
14     System.out.printf("'%c' in uppercase is %c%n", c1, Character.toUpperCase(c1));
15     System.out.printf("'%c' in uppercase is %c%n", c2, Character.toUpperCase(c2));
16   }
17 }
```

Character example (2)

```
1  public class CharacterExample {
2    public static void main(String[] args){
3      char c = 'a';
4      Character c1 = 'A';
5      Character c2 = Character.valueOf(c);
6      System.out.printf("c1 = %c | c2 = %s%n", c1, c2.toString());
7      System.out.printf("c1 and c2 are equal? %b%n", c1.equals(c2));
8      System.out.printf("c1 and c2 are equal when case ignored? %b%n",
9                        c1.toString().equalsIgnoreCase(c2.toString()));
10     System.out.printf("'%c' is digit? %b%n", c1, Character.isDigit(c1));
11     System.out.printf("'%c' is letter? %b%n", c1, Character.isLetter(c1));
12     System.out.printf("'%c' is uppercase? %b%n", c1, Character.isUpperCase(c1));
13     System.out.printf("'%c' is digit? %b%n", c2, Character.isUpperCase(c2));
14     System.out.printf("'%c' in uppercase is %c%n", c1, Character.toUpperCase(c1));
15     System.out.printf("'%c' in uppercase is %c%n", c2, Character.toUpperCase(c2));
16   }
17 }
```

Different initialisations

Character example (3)

```
1  public class CharacterExample {
2    public static void main(String[] args){
3      char c = 'a';
4      Character c1 = 'A';
5      Character c2 = Character.valueOf(c);
6      System.out.printf("c1 = %c | c2 = %s%n", c1, c2.toString());
7      System.out.printf("c1 and c2 are equal? %b%n", c1.equals(c2));
8      System.out.printf("c1 and c2 are equal when case ignored? %b%n",
9                        c1.toString().equalsIgnoreCase(c2.toString()));
10     System.out.printf("'%c' is digit? %b%n", c1, Character.isDigit(c1));
11     System.out.printf("'%c' is letter? %b%n", c1, Character.isLetter(c1));
12     System.out.printf("'%c' is uppercase? %b%n", c1, Charac
13     System.out.printf("'%c' is digit? %b%n", c2, Character.
14     System.out.printf("'%c' in uppercase is %c%n", c1, Char
15     System.out.printf("'%c' in uppercase is %c%n", c2, Char
16   }
17 }
```

```
$ java CharacterExample
c1 = A | c2 = a
c1 and c2 are equal? false
c1 and c2 are equal when case ignored? true
```

Different comparisons; note that `equalsIgnoreCase()` is defined for `String`, but not `Character`

Character example (4)

```
1  public class CharacterExample {
2    public static void main(String[] args){
3      char c = 'a';
4      Character c1 = 'A';
5      Character c2 = Character.valueOf(c);
6      System.out.printf("c1 = %c | c2 = %s%n", c1, c2.toString());
7      System.out.printf("c1 and c2 are equal? %b%n", c1.equals(c2));
8      System.out.printf("c1 and c2 are equal when case ignored? %b%n",
9                        c1.toString().equalsIgnoreCase(c2.toString()));
10     System.out.printf("'%c' is digit? %b%n", c1, Character.isDigit(c1));
11     System.out.printf("'%c' is letter? %b%n", c1, Character.isLetter(c1));
12     System.out.printf("'%c' is uppercase? %b%n", c1, Character.isUpperCase(c1));
13     System.out.printf("'%c' is digit? %b%n", c2, Character.isUpperCase(c2));
14     System.out.printf("'%c' in uppercase is %c%n", c1, Character.toUpperCase(c1));
15     System.out.printf("'%c' in uppercase is %c%n", c2, Character.toUpperCase(c2));
16   }
17 }
```

Basic tests

```
...
'A' digit? false
'A' letter? true
'A' uppercase? true
'a' uppercase? false
```

Character example (5)

```
1  public class CharacterExample {
2    public static void main(String[] args){
3      char c = 'a';
4      Character c1 = 'A';
5      Character c2 = Character.valueOf(c);
6      System.out.printf("c1 = %c | c2 = %s%n", c1, c2.toString());
7      System.out.printf("c1 and c2 are equal? %b%n", c1.equals(c2));
8      System.out.printf("c1 and c2 are equal when case ignored? %b%n",
9                        c1.toString().equalsIgnoreCase(c2.toString()));
10     System.out.printf("'%c' is digit? %b%n", c1, Character.isDigit(c1));
11     System.out.printf("'%c' is letter? %b%n", c1, Character.isLetter(c1));
12     System.out.printf("'%c' is uppercase? %b%n", c1, Character.isUpperCase(c1));
13     System.out.printf("'%c' is digit? %b%n", c2, Character.isUpperCase(c2));
14     System.out.printf("'%c' in uppercase is %c%n", c1, Character.toUpperCase(c1));
15     System.out.printf("'%c' in uppercase is %c%n", c2, Character.toUpperCase(c2));
16   }
17 }
```

Basic character manipulation

```
...
'A' in uppercase is A
'a' in uppercase is A
$
```

Questions, comments?