Character, String and String Builder

Character

- A class whose instances can hold a single character value and whose methods manipulate and inspect single-character data

Character method

- isUpperCase()
  - Tests if character is uppercase
  - Returns true if the argument is uppercase and false otherwise

```
char x = 'A';
System.out.println(Character.isUpperCase(x));

char y = 'e';
System.out.println(Character.isUpperCase(y));
```

```
true
false
```

Character method

- isLowerCase()
  - Tests if character is lowercase
  - Returns true if the argument is lowercase and false otherwise

```
char x = 'A';
System.out.println(Character.isLowerCase(x));

char y = 'e';
System.out.println(Character.isLowerCase(y));
```

```
false
true
```

“Character method

```
char x = ‘b’;

System.out.println(Character.isUpperCase(x));
System.out.println(Character.isLowerCase(x));
```

```
False
True
```

Character method

- toUpperCase()
  - Returns the uppercase equivalent of the argument
  - No change is made if the argument is an uppercase letter

```
char x = 'A';
System.out.println(Character.toUpperCase(x));

char y = 'e';
System.out.println(Character.toUpperCase(y));
```

```
A
E
```

Character method

- toLowerCase()
  - Returns the lowercase equivalent of the argument
  - No change is made if the argument is a lowercase letter

```
char x = 'A';
System.out.println(Character.toLowerCase(x));

char y = 'e';
System.out.println(Character.toLowerCase(y));
```

```
a
e
```

“Character method

```
char x = ‘b’;

System.out.println(Character.toUpperCase(x));
System.out.println(Character.toLowerCase(x));
```

```
B
b
```

Character method

- isDigit()
  - Returns true if the argument is a digit (0–9) and false otherwise
- isLetter()
  - Returns true if the argument is a letter and false otherwise
- isLetterOrDigit()
  - Returns true if the argument is a letter or digit and false otherwise

```
char x = '1';
System.out.println(Character.isDigit(x));
System.out.println(Character.isLetter(x));
System.out.println(Character.isLetterOrDigit(x));
```

```
true
false
true
```

“Character method

```
char x = ‘b’, y = ‘*’;

System.out.println(Character.isDigit(x));
System.out.println(Character.isLetter(x));
System.out.println(Character.isLetterOrDigit(y));
```

```
false
true
false
```

Character method

- isWhitespace()
  - Returns true if the argument is whitespace and false otherwise
  - Includes the space, tab, newline, carriage return, and form feed

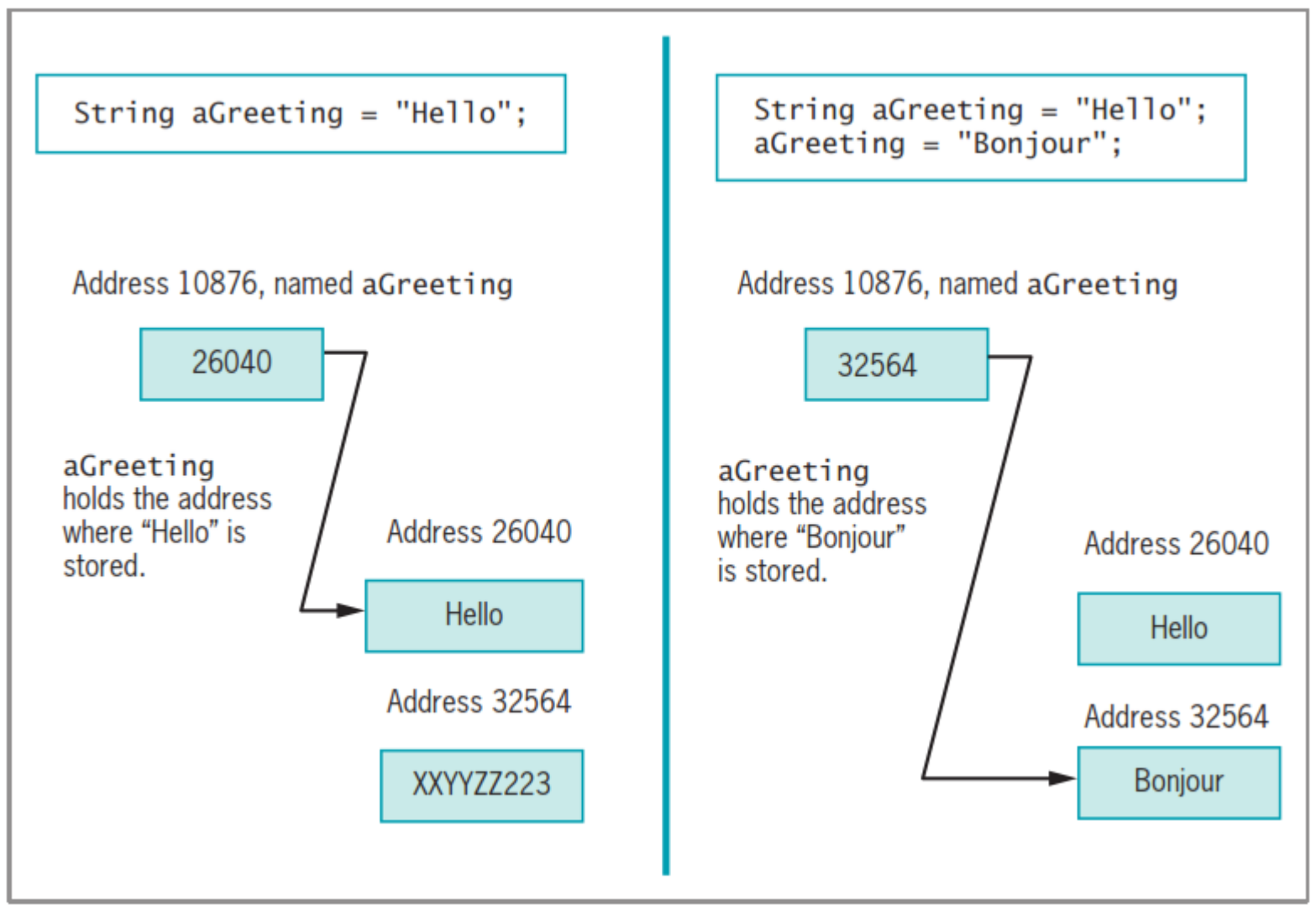
```
char x = ' ';
System.out.println(Character.isWhitespace(x));
```

```
true
```

String

- A class for working with fixed-string data - that is, unchanging data (immutable) composed of multiple characters
- It is a class in Java
  - Each created string is an object
  - String variable name is a reference (memory address)

String


String aGreeting = "Hello";
Address 10876, named aGreeting
26040
aGreeting holds the address where "Hello" is stored.
Address 26040
Hello
Address 32564
XXYYZZ223
String aGreeting = "Hello";
aGreeting = "Bonjour";
Address 10876, named aGreeting
32564
aGreeting holds the address where "Bonjour" is stored.
Address 26040
Hello
Address 32564
Bonjour

String methods

- equals()
  - Evaluates the contents of two String objects to determine if they are equivalent
  - Returns true if the objects have identical contents, no matter how the contents were assigned, and false otherwise

```
Scanner in = new Scanner(System.in);
String x = "Java";
String y = in.next();

System.out.println(x.equals(y));

***if the user typed Java          true
***if the user typed programming   false
***if the user typed java          false
```

String methods

- equalsIgnoreCase()
  - Similar to the equals() method
  - Ignores case when determining if two Strings are equivalent.

```
String x = "Java";
String y = "JaVA";

System.out.println(x.equalsIgnoreCase(y));

                true
```

“String methods

```
String x = "Number";

System.out.println(x.equals("number"));
System.out.println(x.equals("Number"));
System.out.println(x.equalsIgnoreCase("number"));
```

```
false
true
true
```

String method

- compareTo()
  - Used to compare two Strings and returns an integer value.
  - Difference between calling object and the argument.
  - Zero if the two Strings refer to the same value.
  - Negative number if the calling object is less than the argument
  - Positive number if the calling object is more than the argument.

```
Scanner in = new Scanner(System.in);
String x = "Java Programming";
String y = in.nextLine();

System.out.println(x.compareTo(y));

***if the user typed Java programing          -32
***if the user typed Android programming      9
***if the user typed Java Programming         0
***if the user typed Java                     12
```

String method

- compareToIgnoreCase()
  - Similar to the compareTo() method
  - Ignores case when comparing two Strings and returns an integer value.

```
Scanner in = new Scanner(System.in);
String x = "Java Programming";
String y = in.nextLine();

System.out.println(x.compareToIgnoreCase(y));

***if the user typed Java programing          4
***if the user typed java                     12
***if the user typed Java ProgRamming         0
```

“String methods

```
String x = "Number";
String y = "Numbering";
String z = "numbers";

System.out.println(y.compareTo(x));
System.out.println(y.compareTo(z));
System.out.println(x.compareToIgnoreCase(z));
System.out.println(z.compareToIgnoreCase(y));
```

```
3
-32
-1
10
```

String method

- toUpperCase()
  - converts any String to its uppercase equivalent
- toLowerCase()
  - converts any String to its lowercase equivalent

```
String x = “number";

x.toUpperCase();
System.out.println(x);
x = x.toUpperCase();
System.out.println(x);
x = x.toLowerCase();
System.out.println(x);

number
NUMBER
number
```

String method

- length()
  - Returns the length of a String

```
String x = "string methods";
System.out.println(x.length());
```

```
14
```

String method

- indexOf()
  - Determines if a specific character occurs within a String
  - Returns position of the character (starts with 0)
  - Returns –1 if the character does not exist in the String

```
String x = "string methods";

System.out.println(x.indexOf('t'));
System.out.println(x.indexOf('S'));
System.out.println(x.indexOf('e'));
```

```
1
-1
8
```

“String method

```
String x = "object oriented";

x = x.toUpperCase();
System.out.println(x);
x.toLowerCase();
System.out.println(x);

int y = x.length();
System.out.println(x.indexOf('e'));
System.out.println(x.indexOf('T'));
System.out.println(x.indexOf('i'));
System.out.println(x.indexOf(‘O'));
System.out.println(y);
```

```
OBJECT ORIENTED
OBJECT ORIENTED
-1
5
-1
0
15
```

String method

- charAt()
  - Requires an integer argument that indicates the position of the character that the method returns, starting with 0.
  - Error occurs if argument is negative or >= the length of the calling String

```
String x = "string methods";
char y;

System.out.println(x.charAt(5));
y = x.charAt(8);
System.out.println(Character.toUpperCase(y));
```

```
g
E
```

String method

```
String x = “the quick brown fox";
int y;

for(int z = 0; z < x.length(); z++)
       if(x.charAt(z) == ' ')
                y++;

System.out.println(y);
```

3

“String method

```
String x = "jun23@gmail.com";
char y;

for(int i = 0 ; i < x.length(); i++){
    y = x.charAt(i);
    if(y % 4 == 1){
        System.out.println(y);
    }
}

System.out.println("\n");
for(int i = 0 ; i < x.length(); i++){
    y = x.charAt(i);
    if(y % 6 == 3){
        System.out.println(y);
    }
}
```

```
u
m
a
i
m

u
3
i
c
o
```

String method

- endsWith()
  - Takes a String argument and return true or false if a String object does or does not end with the specified argument
  - Case sensitive method

```
String x = "string methods";

System.out.println(x.endsWith("s"));
System.out.println(x.endsWith("ods"));
System.out.println(x.endsWith("oDs"));
```

```
true
true
false
```

String method

- startsWith()
  - Takes a String argument and return true or false if a String object does or does not start with the specified argument
  - Case sensitive method

```
String x = "string methods";

System.out.println(x.startsWith("sT"));
System.out.println(x.startsWith("str"));
System.out.println(x.startsWith("s"));
```

```
false
true
true
```

“String method

```
String x = “object oriented";

System.out.println(x.startsWith(“ob”));
System.out.println(x.startsWith(“ed”));
System.out.println(x.endsWith(“Ted”));
System.out.println(x.endsWith(“rienteD”));
```

```
true
false
false
true
```

"String method

- replace()
  - It replace all occurrences of a character within a String.
  - Case sensitive method
  - No change if the character does not exist within the String.

```
String x = "object oriented",y;

System.out.println(x.replace('o','a'));
System.out.println(x);
y = x.replace('e','i');
System.out.println(y);
System.out.println(x);
```

```
abject ariented
object oriented
objict oriintid
object oriented
```

```
x = x.replace('o','a');
x = x.replace('e','i');
System.out.println(x);
```

```
abjict ariintid
```

String method

- substring()
  - It takes two integer arguments - a start position and an end position (index)
  - The length of the extracted substring is the difference between the second integer and the first integer
  - If you call the method without a second integer argument, the substring extends to the end of the original string.

```
String x = "string methods";

System.out.println(x.substring(2,5));
System.out.println(x.substring(11));
```

```
rin
ods
```

“String methods

```
String x = "object oriented" , y;

y = x.substring(0,6);
System.out.println(y);
System.out.println(x);
x = x.substring(7);
x = x.substring(0,x.length()-2).toUpperCase();
System.out.println(x);
```

```
object
object oriented
ORIENT
```

String method

- trim()
  - Removes trailing spaces in the beginning and end of the String.

```
String x = "      hello      world      ";
System.out.println(x.trim());
String y = "  java program  ";
System.out.println(y.trim());
```

```
hello      world
java program
```

String method

- regionMatches()
  - It can be used to test whether two String regions are the same.
  - One version takes four arguments—the position at which to start in the calling String, the other String being compared, the position to start in the other String, and the length of the comparison.

```
String firstString = "abcde";
String secondString = "xxbcdef";
S.o.p. (firstString.regionMatches(1, secondString, 2, 4));
S.o.p. (firstString.regionMatches(0, secondString, 3, 2));
```

```
true
false
```

String method

- A second version of the regionMatches() method takes an additional boolean argument as the first argument.
- This argument represents whether case should be ignored in deciding whether regions match.

```
String thirdString = "123 Maple Drive";
String fourthString = "a maple tree";
S.o.p. (thirdString.regionMatches(true, 4, fourthString, 2, 5));
```

```
true
```

“String method

```
String x = " programming ";
String y = "programM";
System.out.println(x.regionMatches(0,y,0,4));
x.trim();
System.out.println(x.regionMatches(0,y,0,4));
x = x.trim();
System.out.println(x.regionMatches(1,y,1,5));
System.out.println(x.regionMatches(3,y,3,4));
System.out.println(x.regionMatches(true,3,y,3,4));
```

```
false
false
true
false
true
```

toString

- toString()
  - Converts any object to a string

```
String x;
int y = 4;
double z= 5.65;

x = Integer.toString(y);
System.out.println(x);
x = Double.toString(z);
System.out.println(x);
```

```
4
5.65
```

“Parsing

- Parsing
  - converting a String to a number

```
String x = “649”, y = “123”;

System.out.println(x + y);
S.o.p.(Integer.parseInt(x) + Integer.parseInt(y));
```

```
649123
772
```

```
String x = “152.678”, y = “3.987”;

System.out.println(x + y);
S.o.p.(Double.parseDouble(x) + Double.parseDouble(y));
```

```
152.6783.987
156.665
```

StringBuilder and StringBuffer

- Classes for storing and manipulating changeable data (mutable) composed of multiple characters
- It is used to modify strings without creation of new and different String objects in memory.

Declaration and Instantiation

```
StringBuilder x = new StringBuilder(“Java");

StringBuilder x = new StringBuilder(in.nextLine());

StringBuilder x = null;
x = new StringBuilder(“Java");
```

StringBuilder methods

- capacity()
  - The actual length of the buffer of the StringBuilder object
  - Length of the string + 16
  - Buffer – a memory block which might or might not contain a string. If it does contain a string, the string might not occupy the entire buffer.

```
StringBuilder x = new StringBuilder(“Programming");
int y = x.capacity();
System.out.println(x.length);
System.out.println("Capacity is " + y);
```

```
11
Capacity is 27
```

- setLength()
  - changes the length of a string in a StringBuilder object

```
StringBuilder x = null;
x = new StringBuilder("6311 Hickory Nut Grove Road");
x.setLength(15);
System.out.println(x);
x.setLength(20);
System.out.println(x + "mar");
```

```
6311 Hickory Nu
6311 Hickory Nu-----mar
```

append()

- append()
  - add characters to the end of a StringBuilder object

```
StringBuilder x = new StringBuilder("Happy");
x.append(" birthday");
System.out.println(x);
x.append(" today");
System.out.println(x);
```

```
Happy birthday
Happy birthday today
```

- insert()
  - add characters at a specific location within a StringBuilder object.

```
StringBuilder x = new StringBuilder("Happy");
x.append(" birthday");
System.out.println(x);
x.insert(6, "18th ");
System.out.println(x);
```

```
Happy birthday
Happy 18th birthday
```

- setCharAt()
  - allows you to change a character at a specified position within a StringBuilder object
  - requires two arguments: an integer position and a character.

```
StringBuilder x = new
StringBuilder("Happy");
x.append(" birthday");
System.out.println(x);
x.insert(6, "18th ");
System.out.println(x);
x.setCharAt(6,'2');
System.out.println(x);
```

```
Happy birthday
Happy 18th birthday
Happy 28th birthday
```