S4IC020 - Programmation Orientée Objet (POO)

Cours 5 : nombres et chaînes de caractères

Florian Boudin

Département d'informatique, Université de Nantes

2011–12

Plan

La classe `Number` (1)

- Les types primitifs sont la plupart du temps utilisés lorsque l'on doit travailler avec des nombres :

```
int i = 500;
float gpa = 3.65f;
byte mask = 0xff;
```

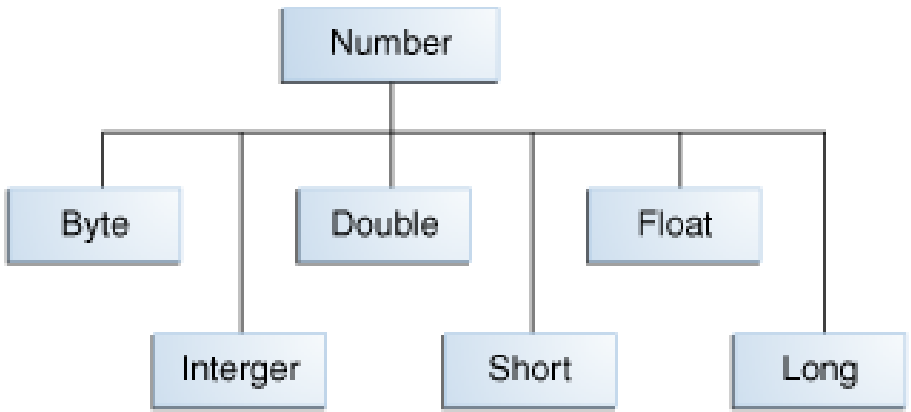
- Il y a cependant des situations où il est préférable d'utiliser des objets, et Java fournit des classes *wrapper* pour chacun des types de données primitifs
- L'*emballage* (*boxing*) est fait par le compilateur si vous utilisez un primitif où un objet est attendu. De même, si vous utilisez un objet `Number` quand un primitif est prévu, le compilateur déballe (*unboxing*) l'objet pour vous

La classe **Number** (2)

- Exemple de *boxing* et *unboxing* :

```
Integer x, y;
x = 12; // boxing
y = 15;
System.out.println(x+y); // unboxing
```

- Toutes les classes *wrapper* sont des sous-classes de la classe abstraite `Number`



Raisons pour utiliser un objet `Number`

1. Comme argument dans une méthode qui prend un objet en paramètre
2. Pour utiliser des constantes définies par la classe, e.g. `MIN_VALUE` et `MAX_VALUE` pour définir les bornes du type de données
3. Pour utiliser les méthodes de classe permettant de convertir les valeurs de et vers d'autres types primitifs, convertir de et vers des `String`, et pour convertir entre systèmes de numération (décimal, octal, binaire, etc.)

Méthodes implémentées par les sous-classes

```
byte byteValue()    // Converts the value of
short shortValue()  // this Number object to
int intValue()      // the primitive data type
long longValue()    // returned
float floatValue()
double doubleValue()

int compareTo(Byte anotherByte)         // Compares this
int compareTo(Double anotherDouble)     // Number object
int compareTo(Float anotherFloat)       // to the argument
int compareTo(Integer anotherInteger)
int compareTo(Long anotherLong)
int compareTo(Short anotherShort)

boolean equals(Object obj)
// return true if the argument is not null and is an
// object of the same type and same numeric value
```

Méthodes de conversion de la classe `Integer`

```
// Decodes a string into an integer
static Integer decode(String s)

// Returns an integer (decimal only)
static int parseInt(String s)

// Returns a String object representing the value
// of this Integer
String toString()

// Returns an Integer object holding the value of
// the specified primitive
static Integer valueOf(int i)
```

- Il existent d'autres méthodes mais elles sont similaires pour toutes les sous-classes de `Number`

Plan

Les méthodes `printf` et `format` (1)

- Le *package* `java.io` contient la classe `PrintStream` qui possède deux méthodes de formatage qui peuvent être utilisées pour remplacer `print` et `println`

```
// Syntaxe de format
PrintStream format(String format, Object... args)
```

- Le premier paramètre est une chaine de caractères spécifiant comment les objets du second paramètre devront être formatés

```
// Exemple
System.out.format("The value of the float " +
      "variable is %f, while the value " +
      "of the integer variable is %d, and " +
      "the string is %s", floatVar, intVar, stringVar);
```

Les méthodes `printf` et `format` (2)

- La chaine de caractère peut contenir du texte brut ainsi que des spécificateurs de format
- Les spécificateurs de format commencent par le signe `%` et se terminent par un convertisseur (caractère indiquant le type d'argument à formater)

```
// Exemple
int i = 461012;
System.out.format("The value of i is: %d%n", i);
// The value of i is: 461012
```

  - `%d` spécifie que la variable est un entier
  - `%n` est un retour à la ligne

- Entre le `%` et le convertisseur, il est possible d'avoir des spécificateurs et drapeaux optionnels

Les méthodes `printf` et `format` (3)

- Les méthodes `printf` et `format` sont surchargées, chacune possède une version avec la syntaxe suivante :

```
public PrintStream format(Locale l,
                          String format,
                          Object... args)
```

- Pour afficher des nombres dans le système français (virgule à la place des points dans les réels), on peut utiliser :

```
System.out.format(Locale.FRANCE,
      "The value of the float variable is %f, " +
      "while the value of the integer variable " +
      "is %d, and the string is %s%n",
      floatVar, intVar, stringVar);
```

Quelques convertisseurs et drapeaux

Converter	Flag	Explication
d		Un entier
f		Un réel (*float*)
s		Une chaîne de caractères
n		Un caractère de retour à la ligne
tB		Une date avec le nom du mois spécifique à la locale
	08	8 caractères de large, avec des zéros si nécessaire
	+	Inclure le signe, positif ou négatif
	-	Justifié à gauche
	.3	3 chiffres après la virgule
	10.3	10 caractères de large, justifié à droite avec 3 chiffres après la virgule

TestFormat (1)

```
import java.util.Locale;

public class TestFormat {
    public static void main(String[] args) {
        long n = 461012;
        System.out.format("%d%n", n);
        System.out.format("%08d%n", n);
        System.out.format("%+8d%n%n", n);

        double pi = 3.14159265;
        System.out.format("%f%n", pi);
        System.out.format("%.3f%n", pi);
        System.out.format("%10.3f%n", pi);
        System.out.format("%-10.3f%n", pi);
        System.out.format(Locale.US, "%-10.4f%n", pi);
    }
}
```

TestFormat (2)

- Sortie de l'exemple précédent

```
461012
00461012
␣+461012

3,141593
3,142
␣␣␣␣␣3,142
3,142
3.1416
```

La classe `DecimalFormat`

- La classe `java.text.DecimalFormat` peut être utilisée pour contrôler l'affichage des zéros (avant, après), des préfixes et suffixes, des séparateurs des milliers et des décimaux

```
String pattern = "###,###.##";
DecimalFormat myFormat = new DecimalFormat(pattern);
String output = myFormat.format(12345.67);
System.out.println(output);
// 12,345.67
```

Plan

La classe `Math` (1)

- La classe `Math` du *package* `java.lang` fournit des méthodes et constantes permettant de réaliser des opérations mathématiques complexes
- Les méthodes de la classe Math sont toutes statiques, vous pouvez les appeler directement à partir de la classe :

```
Math.cos(angle);
```

- Vous pouvez utiliser `import static` afin de ne plus avoir à écrire `Math` au début de chaque fonction mathématique

```
import static java.lang.Math.*;
cos(angle);
```

La classe **Math** (2)

- La classe `Math` contient deux constantes

```
Math.E // base des logarithmes naturels (2,718...)
Math.PI // rapport entre la circonference d'un
        // cercle et son diametre (3,141...)
```

- La classe `Math` contient plus de 40 méthodes statiques

```
double abs(double d)  // Returns the absolute
float abs(float f)    // value of the argument
int abs(int i)
long abs(long lng)

double ceil(double d) // Returns the smallest
                      // integer that is greater
                      // than or equal to the argument
```

La classe **Math** (3)

```
double floor(double d) // Returns the largest
                       // integer that is less
                       // than or equal to the argument

double rint(double d)  // Returns the integer
                       // that is closest in value
                       // to the argument

long round(double d) // Returns the closest long or
int round(float f)   // int to the argument

// Returns the smaller of the two arguments
int min(int arg1, int arg2) // idem double, float et long

// Returns the larger of the two arguments
int max(int arg1, int arg2) // idem double, float et long
```

BasicMathDemo (1)

```
public class BasicMathDemo {
    public static void main(String[] args) {
        double a = -191.635, b = 43.74;
        int c = 16, d = 45;

        System.out.printf("The absolute value of %.3f " +
                          "is %.3f%n", a, Math.abs(a));
        System.out.printf("The ceiling of %.2f " +
                          "is %.0f%n", b, Math.ceil(b));
        System.out.printf("The floor of %.2f " +
                          "is %.0f%n", b, Math.floor(b));
        System.out.printf("The rint of %.2f " +
                          "is %.0f%n", b, Math.rint(b));
        System.out.printf("The max of %d and %d is " +
                          "%d%n", c, d, Math.max(c, d));
        System.out.printf("The min of of %d and %d is " +
                          "%d%n", c, d, Math.min(c, d));
    }
}
```

BasicMathDemo (2)

- Sortie de l'exemple précédent

```
The absolute value of -191,635 is 191,635
The ceiling of 43,74 is 44
The floor of 43,74 is 43
The rint of 43,74 is 44
The max of 16 and 45 is 45
The min of of 16 and 45 is 16
```

Méthodes exponentielles et logarithmiques

```
// Returns the base of the natural logarithms to the
// power of the argument (e^d)
double exp(double d)

// Returns the natural logarithm of the argument
double log(double d)

// Returns the value of the first argument raised
// to the power of the second argument
double pow(double base, double exponent)

// Returns the square root of the argument
double sqrt(double d)
```

ExponentialDemo

```
public class ExponentialDemo {
    public static void main(String[] args) {
        double x = 11.635;
        double y = 2.76;

        System.out.printf("The value of e is %.4f%n",
                          Math.E);
        // The value of e is 2,7183
        System.out.printf("exp(%.3f) is %.3f%n",
                          x, Math.exp(x));
        // exp(11,635) is 112983,831
        System.out.printf("log(%.3f) is %.3f%n",
                          x, Math.log(x));
        // log(11,635) is 2,454
        System.out.printf("pow(%.3f, %.3f) is %.3f%n",
                          x, y, Math.pow(x, y));
        // pow(11,635, 2,760) is 874,008
        System.out.printf("sqrt(%.3f) is %.3f%n",
                          x, Math.sqrt(x));
        // sqrt(11,635) is 3,411
    }
}
```

Méthodes trigonométriques

- La classe `Math` fournit un ensemble de fonctions de calcul trigonométrique, chacune prenant une valeur d'angle exprimée en radians (la méthode `toRadians` convertit les degrés en radians)

```
double sin(double d) // sinus
double cos(double d) // cosinus
double tan(double d) // tangente
double asin(double d) // Arc sinus
double acos(double d) // Arc cosinus
double atan(double d) // Arc tangente
double atan2(double y, double x) // ...
double toDegrees(double d) // convertit en degres
double toRadians(double d) // convertit en radians
```

TrigonometricDemo

```
public class TrigonometricDemo {
    public static void main(String[] args) {
        double degrees = 45.0;
        double radians = Math.toRadians(degrees);

        System.out.format("The value of pi is %.4f%n",
                          Math.PI);
        // The value of pi is 3,1416
        System.out.format("sin(45) is %.4f%n",
                          Math.sin(radians));
        // sin(45) is 0,7071
        System.out.format("cos(45) is %.4f%n",
                          Math.cos(radians));
        // cos(45) is 0,7071
        System.out.format("tan(45) is %.4f%n",
                          Math.tan(radians));
        // tan(45) is 1,0000
    }
}
```

Les nombres aléatoires

- La méthode `random()` retourne un nombre pseudo-aléatoire sélectionné dans $[0.0, 1.0[$
- Pour obtenir un nombre dans un ensemble différent, il faut réaliser une opération arithmétique sur la valeur retournée

```
int number = (int)(Math.random() * 10);
// Entier entre 0 et 9
```

- La méthode `Math.random` fonctionne bien pour générer un nombre aléatoire, la création d'une instance de `java.util.Random` et des invocations aux méthodes seront préférées si vous avez besoin d'une série de nombres aléatoires

Plan

La classe `Character`

- Dans la plupart des cas, vous utiliserez le type primitif `char` pour manipuler des caractères

```
char ch = 'a';
// Unicode for uppercase Greek
// omega character
char uniChar = '\u039A';
// an array of chars
char[] charArray = { 'a', 'b', 'c', 'd', 'e' };
```

- Cependant, comme avec les nombres, il est parfois nécessaire d'utiliser un caractère comme un objet
- Java fournit la classe *wrapper* `Character`

```
Character ch = new Character('a');
Character ch = 'a'; // alternative avec boxing
```

Les méthodes utiles de la classe **Character**

```
boolean isLetter(char ch) // Determine si le caractere
boolean isDigit(char ch)  // est une lettre ou un chiffre

// determine si le caractere est un espace
boolean isWhitespace(char ch)

boolean isUpperCase(char ch) // Determine si le caractere
boolean isLowerCase(char ch) // est majuscule/minuscule

char toUpperCase(char ch) // Retourne le caractere en
char toLowerCase(char ch) // majuscule/minuscule

// Retourne un objet String representant le caractere
toString(char ch)
```

Séquences d'échappement

- Un caractère précédé par une barre oblique inversée (*backslash*) a un comportement spécial pour le compilateur

Séquence	Description
`\t`	Insérer une tabulation
`\b`	Insérer un retour arrière
`\n`	Insérer un retour à la ligne
`\'`	Insérer un guillemet simple
`\"`	Insérer un guillemet double
`\\`	Insérer une barre oblique inversée

```
// Exemple, afficher : She said "Hello!" to me.
System.out.println("She said \"Hello!\" to me.");
```

Plan

Les chaînes de caractères (1)

- Les chaînes de caractères (objets `String` en Java) sont des séquences de caractères
- Pour créer un `String` :

```
String greeting = "Hello world!";
```

- Dans ce cas, `"Hello world!"` est une chaîne de caractères littérale, i.e. une suite de caractères entourés du signe `"`
- Vous pouvez créer un `String` avec le mot-clé `new` :

```
char[] helloArray = {'h', 'e', 'l', 'l', 'o'};
String helloString = new String(helloArray);
System.out.println(helloString);
```

Les chaînes de caractères (2)

- La classe `String` est immutable, i.e. une fois qu'un objet `String` est créé, il ne peut pas être modifié
- Les méthodes de la classe `String` qui semblent pouvoir modifier la chaîne de caractères, créent et retournent un nouvel objet
- La longueur d'une chaîne :

```
String palindrome = "Dot saw I was Tod";
int len = palindrome.length();
```

- Le i^{eme} caractère de la chaîne :

```
// Le premier caractere est en position 0
char a = palindrome.charAt(1);
```

StringDemo

```
public class StringDemo {
    public static void main(String[] args) {
        String palindrome = "Dot saw I was Tod";
        int len = palindrome.length();
        char[] tempCharArray = new char[len];
        char[] charArray = new char[len];

        for (int i = 0; i < len; i++) {  // ?
            tempCharArray[i] = palindrome.charAt(i);
        // palindrome.getChars(0, len, tempCharArray, 0);
        }

        for (int j = 0; j < len; j++) {  // ?
            charArray[j] = tempCharArray[len - 1 - j];
        }

        String reversePalindrome = new String(charArray);
        System.out.println(reversePalindrome);
    }
}
```

Les chaînes de caractères (3)

- La classe `String` contient une méthode pour concaténer deux chaînes de caractères `string1.concat(string2);`

```
"Hello, ".concat("world !");
// est similaire a
"Hello," + " world" + "!"
```

- Java n'autorise pas les chaînes de caractères sur plusieurs lignes, l'opérateur + doit être utilisé en fin de chaque ligne :

```
String quote = "Now is the time for all good " +
        "men to come to the aid of their country.";
```

Plan

Convertir les chaînes en nombres

- Souvent, un programme doit utiliser des données numériques contenues dans une chaîne de caractères
  - e.g. entrée clavier d'un utilisateur, arguments en CLI
- Les sous-classes de `Number` (`Integer`, `Double`, etc.) fournissent la méthode `valueOf` qui convertit une chaîne de caractère en nombre du même type

```
String uneChaine = "16.5";
float a = (Float.valueOf(uneChaine)).floatValue();
// valueOf retourne un objet Float
// floatvalue retourne un type primitif
```

- La méthode `parseXXXX` permet de convertir directement une chaîne de caractères en type primitif :

```
float a = Float.parseFloat(uneChaine);
```

ValueOfDemo

```
public class ValueOfDemo {
    public static void main(String[] args) {
        String s1 = "12";
        String s2 = "3";

        float a = (Integer.valueOf(s1)).intValue();
        float b = Integer.parseInt(s2);

        System.out.println("a + b = " + (a + b));
        System.out.println("a - b = " + (a - b));
        System.out.println("a * b = " + (a * b));
        System.out.println("a % b = " + (a % b));
    }
}
```

Convertir les nombres en chaînes

- Il existe plusieurs méthodes pour convertir un nombre en chaîne de caractères :

```
// Methode 1
int i;
String s1 = "" + i;

// Methode 2
String s2 = String.valueOf(i);

// Methode 3
int i;
double d;
String s3 = Integer.toString(i);
String s4 = Double.toString(d);
```

ToStringDemo

```
public class ToStringDemo {
    public static void main(String[] args) {
        double d = 858.48;
        String s = Double.toString(d);
        int dot = s.indexOf('.');
        // returns the index within this string of the
        // first occurrence of the specified substring

        System.out.println(dot + " digits " +
                           "before decimal point.");
        System.out.println( (s.length() - dot - 1) +
                           " digits after decimal point.");
    }
}
```

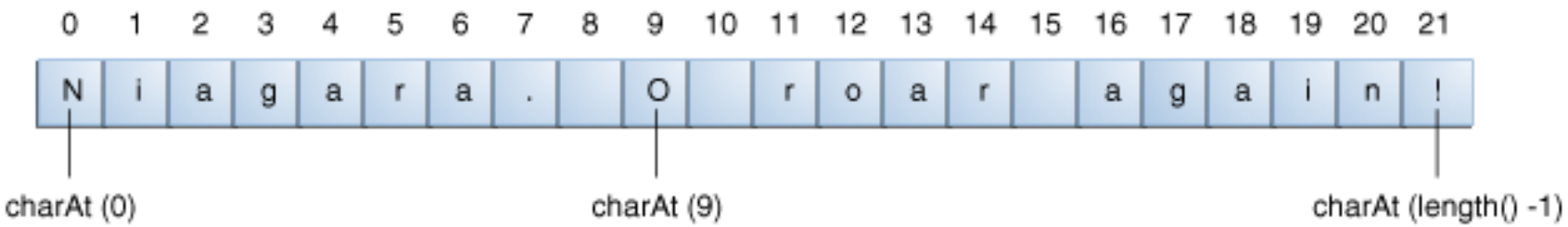
Plan

Manipuler les caractères dans une chaîne (1)

- Extraire un caractère à partir d'une chaîne de caractères :

```
String anotherPalindrome = "Niagara. O roar again!";
char aChar = anotherPalindrome.charAt(9);
```

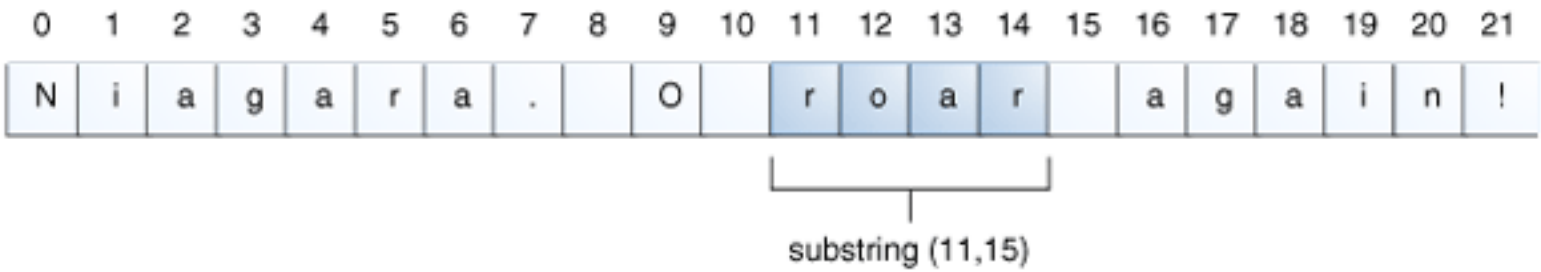
- L'index commence à 0, le caractère à la position 9 est `'O'`, comme illustré par la figure ci-dessous :



Manipuler les caractères dans une chaîne (2)

- Extraire une sous-chaîne de caractères avec la méthode `substring` :

```
String anotherPalindrome = "Niagara. O roar again!";
String roar = anotherPalindrome.substring(11, 15);
```



- Il existe deux versions de la méthode

```
String substring(int beginIndex, int endIndex)
String substring(int beginIndex)
```

Manipuler les caractères dans une chaîne (3)

- Quelques autres méthodes utiles :

```
String[] split(String regex)
String[] split(String regex, int limit)
// recherche les occurrences de regex pour decouper
// la chaine et la mettre dans un tableau

CharSequence subSequence(int beginIndex, int endIndex)
// retourne une sous-sequence de caracteres

String trim()
// retourne une copie de la chaine avec les espaces
// en debut/fin de chaine supprimes

String toLowerCase()
String toUpperCase()
// Retourne une copie de la chaine en
// majuscules / minuscules
```

Rechercher des caractères dans une chaîne

```
int indexOf(String str)
int lastIndexOf(String str)
// retourne l'index de la premiere/derniere occurrence
// de la sous-chaine

int indexOf(String str, int fromIndex)
int lastIndexOf(String str, int fromIndex)
// retourne l'index de la premiere/derniere occurrence
// de la sous-chaine en recherchant a partir d'un index

boolean contains(CharSequence s)
// retourne vrai si la chaine contient la sequence de
// caracteres
```

Remplacer des caractères dans une chaîne

```
String replace(char oldChar, char newChar)
// retourne un nouveau String ou les occurrences
// de oldChar sont remplacees par newChar

String replace(CharSequence target, CharSequence replace)
// remplace chaque sequence de caracteres target

String replaceAll(String regex, String replace)
// remplace toutes les sous-chaines regex par replace

String replaceFirst(String regex, String replace)
// remplace la premiere occurrence de regex par replace
```

FilenameDemo

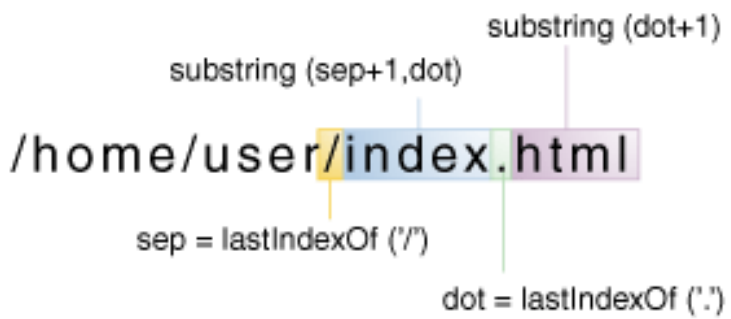
```
public class FilenameDemo {
    public static void main(String[] args) {
        final String path = "/home/user/index.html";
        Filename page = new Filename(path, '/', '.');

        System.out.println( "Extension = " +
                            page.extension() );
        System.out.println( "Filename = " +
                           page.filename() );
        System.out.println( "Path = " + page.path() );
    }
}
// Informations a partir du chemin d'un fichier
// Extension = html
// Filename = index
// Path = /home/mem
```

Filename - questions

```
public class Filename {
    String fullPath;
    char pathSeparator, extensionSeparator;

    public Filename(String str, char sep, char ext) {
        fullPath = str;
        pathSeparator = sep;
        extensionSeparator = ext;
    }

    public String extension() // ...

    public String filename() // ...

    public String path() // ...
}
```

Filename - réponses

```
public String extension() {
    int dot = fullPath.lastIndexOf(extensionSeparator);
    return fullPath.substring(dot + 1);
}

public String filename() {
    int dot = fullPath.lastIndexOf(extensionSeparator);
    int sep = fullPath.lastIndexOf(pathSeparator);
    return fullPath.substring(sep + 1, dot);
}

public String path() {
    int sep = fullPath.lastIndexOf(pathSeparator);
    return fullPath.substring(0, sep);
}
```



Comparaison de chaînes de caractères (1)

```
boolean endsWith(String suffix)
boolean startsWith(String prefix)
boolean startsWith(String prefix, int offset)
// retourne vrai si la chaine commence/termine par la
// chaine specifiee en parametre (offset possible)

int compareTo(String anotherString)
int compareToIgnoreCase(String str)
// Compare deux chaines lexicographiquement
"abc".compareTo("bcd") // -1
"abc".compareTo("abc") // 0
"abc".compareTo("aab") // +1

boolean equals(Object anObject)
boolean equalsIgnoreCase(String anotherString)
// retourne vrai si les deux chaines contiennent
// la meme sequence de caracteres
```

Comparaison de chaînes de caractères (2)

```
boolean regionMatches(int toffset, String other,
                      int ooffset, int len)
boolean regionMatches(boolean ignoreCase, int toffset,
                      String other, int ooffset, int len)
// test si la region specifiee matche la region
// specifiee de la chaine en parametre
"abcd".regionMatches(0, "bc", 0, 2) // false
"abcd".regionMatches(1, "bc", 0, 2) // true

boolean matches(String regex)
// test si cette chaine matche la regex
```

RegionMatchesDemo

```
public class RegionMatchesDemo {
    public static void main(String[] args) {
        String searchMe = "Green Eggs and Ham";
        String findMe = "Eggs";
        int searchMeLen = searchMe.length();
        int findMeLen = findMe.length();
        boolean foundIt = false;

        for (int i = 0; i <= (searchMeLen - findMeLen); i++) {
            if (searchMe.regionMatches(i, findMe, 0, findMeLen)) {
                foundIt = true;
                System.out.println(
                    searchMe.substring(i, i + findMeLen));
                break;
            }
        }

        if (!foundIt)
            System.out.println("No match found.");
    }
}
```

Plan

La classe StringBuilder (1)

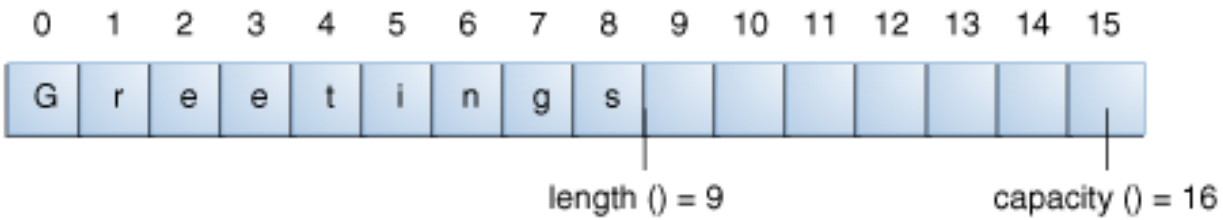
- Les objets `StringBuilder` sont similaires aux `String` à la différence qu'ils peuvent être modifiés
- La longueur et le contenu de la sequence de caractères peuvent être modifiés par le biais de méthodes
- Comme pour `String`, la méthode length() retourne la longueur de la sequence de caractères
- Chaque objet `StringBuilder` possède une capacité, i.e. le nombre de caractères qu'il peut contenir

La classe StringBuilder (2)

```
StringBuilder()
// creer un objet StringBuilder vide avec une
// capacite de 16 elements

StringBuilder(CharSequence cs)
// construit un objet StringBuilder avec les
// caracteres plus 16 elements vides

StringBuilder(int initCapacity)
// creer un objet StringBuilder vide avec la
// capacite initialisee par le parametre

StringBuilder(String s)
// creer un un objet StringBuilder avec la
// chaine plus 16 elements vides
```

La classe StringBuilder (3)

```
StringBuilder sb = new StringBuilder();
// creer un StringBuilder vide de capacite
// egale a 16 elements

sb.append("Greetings");
// ajoute une chaine de 9 caracteres au debut
```




```
void setLength(int newLength)
// fixe la longueur de la sequence de caracteres

void ensureCapacity(int minCapacity)
// veille a ce que la capacite soit au moins egal
// au minimum specifie
```

Les opérations avec StringBuilder

```
StringBuilder append(int i)
StringBuilder append(String s) // ...
// ajoute le parametre a l'objet StringBuilder

StringBuilder delete(int start, int end)
StringBuilder deleteCharAt(int index)
// supprime un caractere/une plage de caracteres

StringBuilder insert(int offset, int i)
StringBuilder insert(int offset, String s) // ...
// insere le parametre dans l'objet StringBuilder

StringBuilder replace(int start, int end, String s)
void setCharAt(int index, char c)
// remplace le/les caractere(s) specifie(s)

StringBuilder reverse() // inverse les caracteres
```

StringBuilderDemo

```
public class StringBuilderDemo {
    public static void main(String[] args) {
        String palindrome = "Dot saw I was Tod";

        StringBuilder sb = new StringBuilder(palindrome);

        sb.reverse();  // inverser la chaine

        System.out.println(sb);
    }
}
```

Plan

Questions

1. Quelles est la capacité initiale de `sb` ?

```
StringBuilder sb = new StringBuilder("Able was I ere I saw Elba.");
```

2. Sachant la chaîne de caractères suivante :

```
String hannah = "Did Hannah see bees? Hannah did.";
```

   a. Quelle est la valeur retournée par `hannah.length()` ?
   b. Quelle est la valeur retournée par `hannah.charAt(12)` ?
   c. Écrire une expression qui se réfère à la lettre b.

3. Quelle est la chaine retournée par l'expression suivante ?

```
"Was it a car or a cat I saw?".substring(9, 12)
```

4. Quelle est la valeur de `result` après chaque ligne ?

```
String original = "software";
StringBuilder result = new StringBuilder("hi");
int index = original.indexOf('a');
result.setCharAt(0, original.charAt(0));
result.setCharAt(1, original.charAt(original.length()-1));
result.insert(1, original.charAt(4));
result.append(original.substring(1,4));
result.insert(3, (original.substring(index, index+2)+" "));
```

Réponses

1. Quelles est la capacité initiale de `sb` ? 26 + 16 = 42

```
StringBuilder sb = new StringBuilder("Able was I ere I saw Elba.");
```

2. Sachant la chaîne de caractères suivante :

```
String hannah = "Did Hannah see bees? Hannah did.";
```

   a. Quelle est la valeur retournée par `hannah.length()` ? 32
   b. Quelle est la valeur retournée par `hannah.charAt(12)` ? e
   c. Écrire une expression qui se réfère à la lettre b. `hannah.charAt(15)`

3. Quelle est la chaine retournée par l'expression suivante ? car

```
"Was it a car or a cat I saw?".substring(9, 12)
```

4. Quelle est la valeur de `result` après chaque ligne ?

```
String original = "software";
StringBuilder result = new StringBuilder("hi");
int index = original.indexOf('a');
result.setCharAt(0, original.charAt(0)); // si
result.setCharAt(1, original.charAt(original.length()-1)); // se
result.insert(1, original.charAt(4)); // swe
result.append(original.substring(1,4)); // sweoft
result.insert(3, (original.substring(index, index+2)+" ")); // swear oft
```