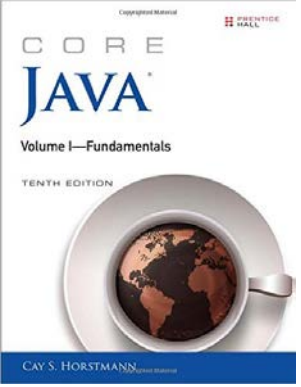


Fundamental Programming Structures in Java – Part 2

Chapter 3, Core Java, Volume I

Contents

- A Sample Program in Java
- Data Types
- Variables
- Type Conversions
- Strings
- Input and Output
- Operators
- Control Flows
- Methods
- Class Structure
- Arrays
- Command-Line Parameters

Strings

- Sequences of Unicode characters.
- String literals enclosed in double quotes: "Java\u2122" denotes Java™
- Java does not have a built-in string type.
- Instead, Java standard library contains a predefined class called String.
- A quoted string is an *instance (object)* of the String class.

```
String e = "";
String greeting = "Hello";
```

- Substrings

```
String greeting = "Hello";
String s = greeting.substring(0, 3);
```

  - Positions start at zero (0 ~ length-1)
  - Last position is excluded—**s.substring(a, b)** has length b - a.

Strings

- Concatenation (+) joins strings:

```
String expletive = "Expletive";
String PG13 = "deleted";
String message = expletive + PG13;  // "Expletivedeleted"
```

- If one operand is not a string, it is turned into a string:

```
int age = 13;
String rating = "PG" + age;
```

- String comparison:

```
"Hello".equals(greeting);
"Hello".equalsIgnoreCase("hello"); // ignoring the case
```

- Caution: Do not use the == operator.

```
"Hello"==greeting;  // probably true  if greeting referes to "Hello"
"Hello".substring(0, 3) == "Hel"; // probably false
```

- Empty string "" has length 0 and is different from *null* string

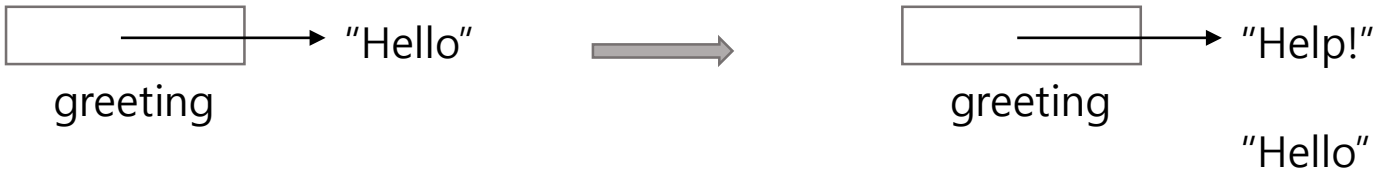
Strings

- Strings are *immutable*
  - no way to change in an existing string itself

- Modifying a string variable

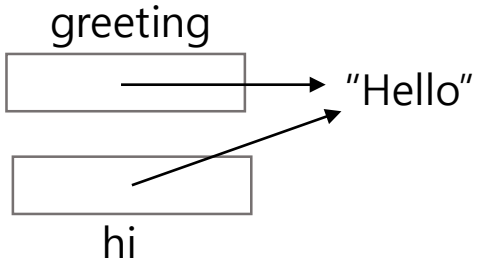
```
String greeting = "Hello";
greeting = greeting.substring(0,3)+'p!";
```

➔does not change the "Hello" string, but changes the value of the greeting variable



- Copying a string variable

```
String hi = greeting;
```



Code Points and Code Units

- s.length() is the number of code units (not Unicode characters).

```
String greeting = "Hello";
int n = greeting.length();  // is 5
```

- To get the true length – the number of code points (Unicode characters)

```
int cpCount = greeting.codePointCount(0, greeting.length());
```

- s.charAt(i) is the i-th code unit.

```
char first = greeting.charAt(0);                    // 'H'
char last = greeting.charAt(greeting.length()-1);  // 'o'
```

- To get the i-th code point:

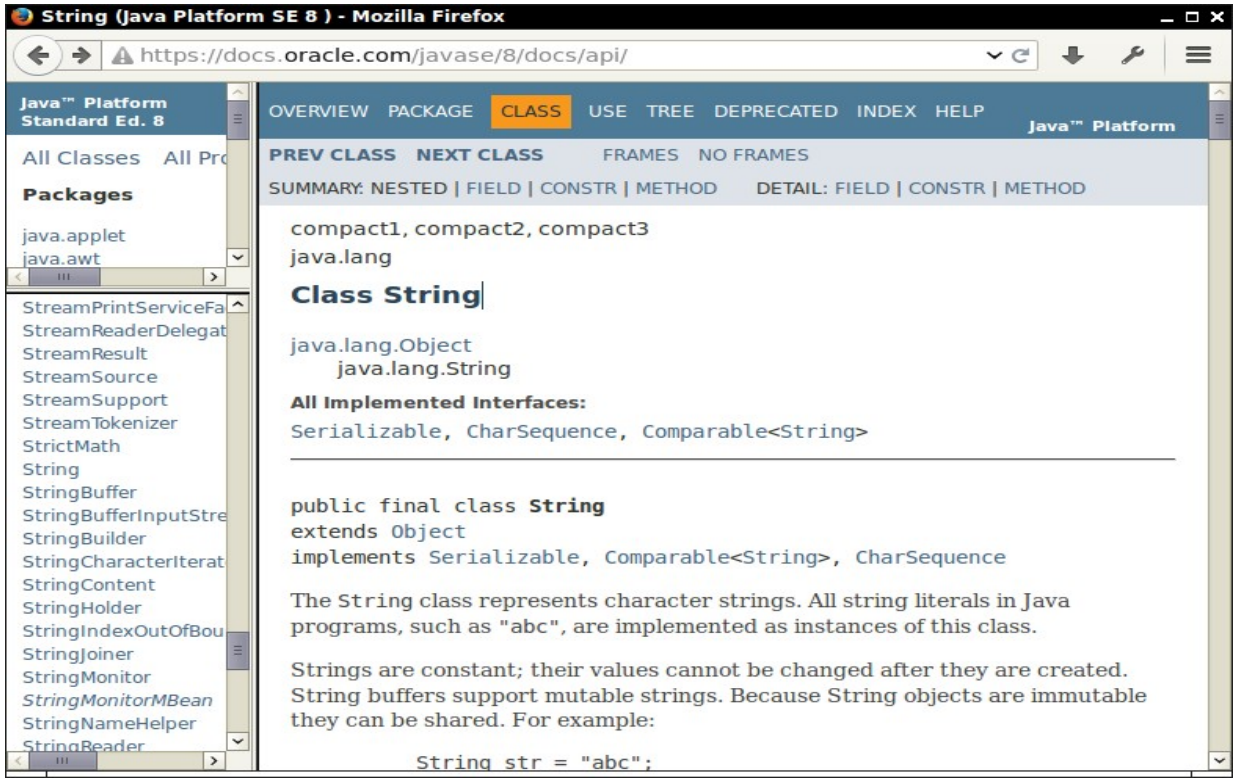
```
int index = s.offsetByCodePoints(0, i);
int cp = s.codePointAt(index);  // index denotes the index of code units
```

- To get all code points:

```
int[] codePoints = s.codePoints().toArray();
```

The String API

- String class in Java contains more than 50 methods.
- Check out the online API documentation!
- API documentation is part of the JDK (docs/api/index.html)



Input and Output

- Standard output stream object (console window)
  - System.out
  - System.out.print, System.out.println, etc.
- Standard input stream object (keyboard)
  - System.in
- Easiest way to read text input from the console : java.util.Scanner class

```
Scanner in = new Scanner(System.in);
System.out.print("What is your name?");
String name = in.nextLine();   // method call
```

  - To read numbers:
    - nextInt()
    - netxDouble()
  - Need to add import statement:

```
import java.util.*;
```

Input Test

```
import java.util.*;

public class InputTest
{
   public static void main(String[] args)
   {
      Scanner in = new Scanner(System.in);

      // get first input
      System.out.print("What is your name? ");
      String name = in.nextLine();

      // get second input
      System.out.print("How old are you? ");
      int age = in.nextInt();

      // display output on console
      System.out.println("Hello, " + name + ". Next year, you'll be " + (age + 1));
   }
}
```

Formatted Output

- Use `printf` for formatted printing:

```
System.out.printf("Price:%8.2f", 10000.0 / 3.0); // Prints Price: 3333.33
```

- Conversion characters for `printf` (table 3.5)
  - Use `f` for floating-point, `d` for integer, `s` for strings and other objects.

- Flags modify output (table 3.6):

```
System.out.printf("%(,.2f", -10000.0 / 3.0); // prints (3,333.33)
```

- Use `String.format` to make a formatted string without printing:

```
String message = String.format("Hello, %s. Next year, you'll be %d", name, age);
```

File Input and Output

- Reading from a text file:

```
Scanner in = new Scanner( "in.txt");
in.nextInt();
```

- Writing to a text file:

```
PrintWriter out = new PrintWriter("out.txt");
out.println(…);
```

- Standard I/O Redirection

```
java MyProg < in.txt > out.txt
```

Operators

Operators	Associativity
[] . () (method call)	Left-to-right
! ~ ++ -- +(unary) –(unary) () (cast) new	Right-to-left
* / %	Left-to-right
+ -	Left-to-right
<< >> >>>	Left-to-right
< <= > >= instanceof	Left-to-right
== !=	Left-to-right
&	Left-to-right
^	Left-to-right
|	Left-to-right
&&	Left-to-right
||	Left-to-right
?:	Right-to-left
= += -= *= /= %= &= |= ^= <<= >>= >>>=	Right-to-left

Control Flow

- if statements
- switch statements
  - case label : string literal (Java 7)
- while and do/while statements
- for statements
  - for-each loop (enhanced-for)

for(type variable: collection) statement

```
int[] numbers = {3, 4, 5, -5, 0, 12};
int sum = 0;
for (int number: numbers)
{
   sum += number;
}
```

```
String s = …;
switch (s)
{
  case "yes" :
    …
    break;
  case "no" :
    …
    break;
   …
}
```

Methods (Functions)

- Static Methods
  - Can be called without creating objects
- Defining and calling static methods

```
/*
 * compute binomial coefficient
   n*(n-1)*(n-2)*...*(n-k+1)/(1*2*3*...*k)
*/
```

```
public class LotteryOdds
{
  public static void main(String[] args)
  {
    ...
    lotteryOdds = odds(n, k); //call a method
    ...
  }


  static int odds(int n, int k)
  {
    int lotteryOdds = 1;
    for (int i = 1; i <= k; i++)
      lotteryOdds = lotteryOdds * (n - i + 1) / i;
    return lotteryOdds;
  }
}
```

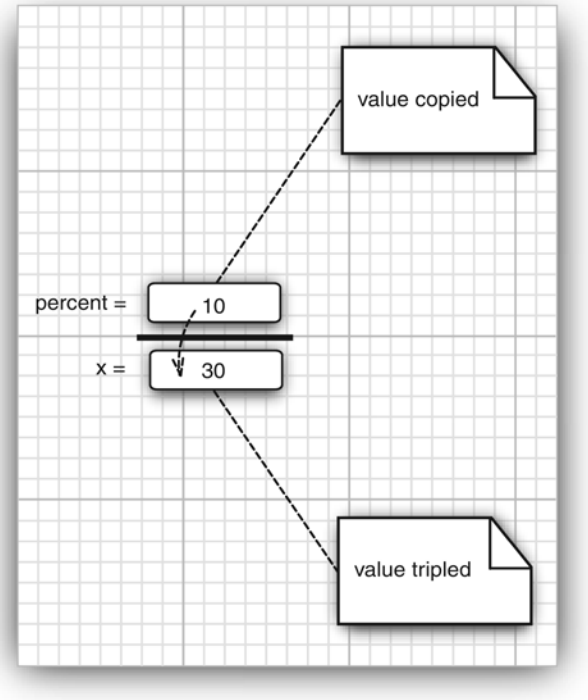
Parameter Passing

- Call by value: The method gets copies of the argument values.
- A method cannot change the contents of variables passed to it.
- Example:

```
public static void tripleValue(double x) // doesn't work
{
   x = 3 * x;
}
```

- In the following call, the percent variable is not changed:

```
double percent = 10;
tripleValue(percent);
```



Class Math and Mathematical Functions

- The **Math** class contains an assortment of mathematical functions.
  - Trigonometric functions : **sin**, **cos**, **tan**, **atan**, **toRadians**, **toDegrees**, …
  - Exponential and logarithm functions : **exp**, **log**, **log10**, …
  - Manipulating values : **abs**, **round**, …
  - Maximum and minimum values : **max**, **min**
  - Square root : **sqrt**
  - Random number : **random**
  - …
- Calling Methods (static methods)
  - `Math.sqrt(4.0)`
  - `Math.cos(Math.toRadians(290))`
- Mathematical Constants (declared with static final)
  - `Math.PI` // π
  - `Math.E` // *e*

Class Structure with Static Methods and Data

- A class contains static methods and variables.
- One of the static methods is the main method.
- A static method can call another static methods.
- Static variables are shared among all the static methods.

```
public class class_name
{
    static variable declaration-1
    static variable declaration-2
    ...
    static variable declaration-k

    static method-1  (main method)
    static method-2
    ...
    static method-n
}
```

- This is not the standard class structure for object-oriented programming, but you can use structured programming with this structure.

Example: Stack Class with Static Methods

- Non-Object-Oriented Stack

```
public class Stack
{
   private static final int MAX = 10;
   private static int top=0;
   private static int[] s = new int[MAX];
   public static int pop()
   {
       if (top == 0)  {
           System.out.println("Empty!");
           System.exit(-1);
       } else  {    top--;
           return s[top];
       }
   }
   public static void push(int x)
   {
     if(top==MAX) {
        System.out.println("Full!");
        System.exit(-1);
     } else  {
        s[top] = x;
        return;
     }
   }
```

```
   public static main(String[] args)
   {
       push(1);
       push(2);
       push(3);
       System.out.println( pop() );
       System.out.println( pop() );
       System.out.println( pop() );
   }
} // end of Stack
```

Example: Stack Class with Static Methods

- Mutiple Classes

```
public class Stack
{
   private static final int MAX = 10;
   private static int top=0;
   private static int[] s = new int[MAX];
   public static int pop()
   {
       if (top == 0)  {
           System.out.println("Empty!");
           System.exit(-1);
       } else  {    top--;
           return s[top];
       }
   }
   public static void push(int x)
   {
     if(top==MAX) {
        System.out.println("Full!");
        System.exit(-1);
     } else  {
        s[top] = x;
        return;
     }
   }
}
```

Main method in separate Java File from Stack.java

```
// StackTest.java
public class StackTest
   public static main(String[] args)
   {
       Stack.push(1);
       Stack.push(2);
       Stack.push(3);
       System.out.println (Stack.pop() );
       System.out.println( Stack.pop() );
       System.out.println( Stack.pop() );

       Stack.s[5] = 10;  // error, not public
   }
} // end of Stack
```