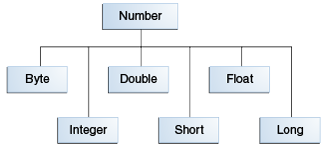
**Numbers and Strings**

* **Numbers**
* **The Number Classes**
  + The java platform provides wrapper classes for each of the primitive data types.
  + These classes wrap the primitive in an Object.
  + The wrapping is done by compiler.
  + If you use a primitive where an object is expected, the compiler boxes the primitive in its wrapper classes for you.
  + If you use a number object when a primitive is expected, the compiler unboxes the object for you.
  + All of the numeric wrapper classes are subclasses of the abstract class Number.



* + There are four other subclasses of Number.
    - BigDecimal
    - BigInterger
    - AtomicInterger
    - AtomicLong
  + The following table list the instance methods that all the subclasses of the Number class implement.

|  |  |
| --- | --- |
| **Methods Implemented by all Subclasses of Number** | |
| **Method** | **Description** |
| byte byteValue() short shortValue() int intValue() long longValue() float floatValue() double doubleValue() | Converts the value of this Number object to the primitive data type returned. |
| int compareTo(Byte anotherByte) int compareTo(Double anotherDouble) int compareTo(Float anotherFloat) int compareTo(Integer anotherInteger) int compareTo(Long anotherLong) int compareTo(Short anotherShort) | Compares this Number object to the argument. |
| boolean equals(Object obj) | Determines whether this number object is equal to the argument. The methods return true if the argument is not null and is an object of the same type and with the same numeric value. There are some extra requirements for Double and Float objects that are described in the Java API documentation |

|  |  |
| --- | --- |
| **Conversion Methods, Integer Class** | |
| **Method** | **Description** |
| static Integer decode(String s) | Decodes a string into an integer. Can accept string representations of decimal, octal, or hexadecimal numbers as input. |
| static int parseInt(String s) | Returns an integer (decimal only). |
| static int parseInt(String s, int radix) | Returns an integer, given a string representation of decimal, binary, octal, or hexadecimal (radix equals 10, 2, 8, or 16 respectively) numbers as input. |
| String toString() | Returns a String object representing the value of this Integer. |
| static String toString(int i) | Returns a String object representing the specified integer. |
| static Integer valueOf(int i) | Returns an Integer object holding the value of the specified primitive. |
| static Integer valueOf(String s) | Returns an Integer object holding the value of the specified string representation. |
| static Integer valueOf(String s, int radix) | Returns an Integer object holding the integer value of the specified string representation, parsed with the value of radix. For example, if s = "333" and radix = 8, the method returns the base-ten integer equivalent of the octal number 333. |

* **Formatting Numeric Print Output**
  + The java.io package includes a PrintStream class that has two formatting methods format() and printf().
  + You can invoke PrintStream methods on System.out
  + The syntax for these two java.io.PrintStream method is same.

*public PrintStream format(String format, Object…args);*

* + The first parameter format is a format string specifying how the objects in the second parameter args are to be formatted.
  + The format string contains plain text as well as format specifier.
  + The format specifier begin with a percent sign(%) and ends with a converter.
  + The converter is a character indicating the type of argument to be formatted.

int i = 461012;

System.out.format("The value of i is: %d%n", i);

* + **The DecimalFormat Class** 
    - You can use the java.text.DecimalFormat class to control the display of leading and trailing zeros, prefixes and suffixes, grouping separators and the decimal separator.
* **Beyond Basic Arithmetic**
  + The Math class in java.lang package provides methods and constants for doing more advanced mathematical computation.
  + The methods in Math class are all static, you can call them directly from the class like this, *Math.cos(angle)*
  + Using the static import language feature, you don’t have to write Math in front of every math function. import static java.lang.Math.\*
  + Constants and Basic Methods
    - The Math class includes two constants. Math.E and Math.PI

| **Basic Math Methods** | |
| --- | --- |
| **Method** | **Description** |
| double abs(double d) float abs(float f) int abs(int i) long abs(long lng) | Returns the absolute value of the argument. |
| double ceil(double d) | Returns the smallest integer that is greater than or equal to the argument. Returned as a double. |
| double floor(double d) | Returns the largest integer that is less than or equal to the argument. Returned as a double. |
| double rint(double d) | Returns the integer that is closest in value to the argument. Returned as a double. |
| long round(double d) int round(float f) | Returns the closest long or int, as indicated by the method's return type, to the argument. |
| double min(double arg1, double arg2) float min(float arg1, float arg2) int min(int arg1, int arg2) long min(long arg1, long arg2) | Returns the smaller of the two arguments. |
| double max(double arg1, double arg2) float max(float arg1, float arg2) int max(int arg1, int arg2) long max(long arg1, long arg2) | Returns the larger of the two arguments. |

* + **Random Numbers**
    - The random() method returns a pseudo randomly selected number between 0.0 and 1.0
    - The range includes 0.0 but not 1.0
    - To generate an integer between 0 and 9, you would write
    - int number = (int) (Math.random() \* 10)
* **Characters**
  + Most of the time if you are using a single character value, you will use primitive char type. Eg char ch = ‘a’;
  + When you need to use a char as an object, for example a method argument where an object is expected.
  + The java programming language provides the wrapper class that wraps the char in a character object for this purpose.
  + You can create a Character object with the Character constructor.

*Character ch = new Character (‘a’)*

* + The java compiler automatically converts the char to Character. This feature is called Autoboxing. If the conversion goes to other way is called unboxing.
  + The Character class is immutable, so that once it’s created, a Character object cannot be changed.

|  |  |
| --- | --- |
| **Useful Methods in the Character Class** | |
| **Method** | **Description** |
| boolean isLetter(char ch) boolean isDigit(char ch) | Determines whether the specified char value is a letter or a digit, respectively. |
| boolean isWhitespace(char ch) | Determines whether the specified char value is white space. |
| boolean isUpperCase(char ch) boolean isLowerCase(char ch) | Determines whether the specified char value is uppercase or lowercase, respectively. |
| char toUpperCase(char ch) char toLowerCase(char ch) | Returns the uppercase or lowercase form of the specified char value. |
| toString(char ch) | Returns a String object representing the specified character value — that is, a one-character string. |

* **Escape Sequences**
  + A character preceded by backslash (\) is an escape sequence and has special meaning to the compiler.
  + When an escape sequence is encountered in print statement, the compiler interprets it’s accordingly.

|  |  |
| --- | --- |
| * **Escape Sequences** | |
| **Escape Sequence** | **Description** |
| \t | Insert a tab in the text at this point. |
| \b | Insert a backspace in the text at this point. |
| \n | Insert a newline in the text at this point. |
| \r | Insert a carriage return in the text at this point. |
| \f | Insert a formfeed in the text at this point. |
| \' | Insert a single quote character in the text at this point. |
| \" | Insert a double quote character in the text at this point. |
| \\ | Insert a backslash character in the text at this point. |

* **Strings**
  + Strings, which are widely used in java programming language, are a sequence of characters.
  + In java programming language strings are objects.
  + The java platform provides the String class to create and manipulate the strings.
* **Creating Strings**
  + The most direct way to create the string is to write

*String greetings = “Hello Word”;*

* + String literal is serious of a character that’s enclosed in double quotes.
  + As with any other object, you can create string object using new keyword and constructors.
  + String class has thirteen constructors that allow you to provide the initial value of string using different sources.

char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };

String helloString = new String(helloArray);

System.out.println (helloString);

* **String Length**
  + Methods used to obtain information about object are known as accessor methods.
  + The length() method returns the number of characters contained in the string object.

stringVar.length();

* **Concatenating Strings**
  + The string class includes the methods for concatenating two strings.

string1.concat(string2);

* + You can also use concat method with string literals.

“My name is”.concat(“Regu”);

* + Strings are more commonly concatenated with + operator.

“My Name is” + “Regu”;

* + + operator is widely used in print statements.
* **Creating format Strings**
  + String class has an equivalent class method format (), that returns the string object rather than a print stream object.
  + Using String’s static format() method allows you to create a formatted string that you can reuse.

String fs;

fs = String.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);

System.out.println(fs);

* **Converting Between Numbers and Strings**
  + **Converting Strings to Numbers**
    - The Number subclasses (Integer, Float, Byte, Double, Long and Short) provide a class method valueOf (), that converts a string to an object of that type.

*String str = “10.20”;*

*float a = (Float.valueOf (str)).floatValue();*

* + - Each Number sub classes also provides a parseXXX () method, that can be used to convert string to primitive numbers.

*float a = Float.parseFloat(str);*

* + **Converting Numbers to Strings**
    - There are several easy ways to convert a number to a string.
      * Concatenate primitive numbers with empty string

int i;

String s1 = “” + i;

* + - * The valueOf() class method.

String s2 = String.valueOf (i);

* + - * Each of the number subclasses includes a class method toString(), that will convert its primitive type to a string.

String s3 = Integer.toString (i);

* **Manipulating Characters in a String**
  + **Getting Characters and Substrings by Index**
    - You can get the character at a particular index within a string by invoking the chatAt() accessor method.
    - The index of the first character in 0, while the index of the last character is length-1

*String str = “Hello World”;*

*Char c = str.charAt(4);*

* + - If you want to get more than one consecutive character from a string, you can use the substring() method.
    - The substring() method has two versions.

*String substring(int beginIndex, int endIndex);*

*String substring(int beginIndex)*

*String res = str.substring(2, 5);*

* + **Other Methods for Manipulating Strings**
    - *String[] split(String regex, int limit)* – Searches for a match as specified by the string argument and split this string into an array of strings accordingly. The optional integer argument specifies the maximum size of the returned array.
    - *CharSequence subsequence(int beginIndex, int endIndex)* – Return new character sequence constructed from beginIndex index up until endindex-1
    - *String trim() –* Returns *a* copy of this string with leading and trailing whitespace removed.
    - *String toLowerCase()* – Returns a copy of this string converted to lower case.
    - String toUpperCase() – Returns a copy of this string converted to upper case.

* + **Searching for Characters and Substrings in a String**
    - *int indexOf(int ch, int fromIndex)* – Returns the index of the first occurrence of the specified character searching forward from the specified index.
    - *int lastIndexOf(int ch, int fromIndex)* – Returns the index of the last occurrence of the specified character, searching backward from the specified index.
    - *int indexOf(String str, int fromIndex)* – Returns the index of the first occurrence of the specified substring searching forward from the specified index.
    - *int lastIndexOf(String str, int fromIndex)* – Returns the index of the last occurrence of the specified substring searching backward from the specified index.
    - *boolean contains(CharSequence s)* – Returns true if the string contains the specified character sequence.
  + **Replacing Characters and Substrings into a String**
    - *String replace(char oldChar, char newChar)* – Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
    - String replace(charSequence target, charSequence replacement) – Replace each substring of this string that matches the literal target sequence with the specified literal replacement sequence.
    - *String replaceAll(String regex, String replacement)* – Replaces each substring of this string that matches the given regular expression with the given replacement.
    - *String replaceFirst(String regex, String replacement)* – Replaces the first substring of this string that matches the given regular expression with the given replacement.
* **Comparing Strings and Portions of Stings**

|  |  |
| --- | --- |
| **Methods for Comparing Strings** | |
| **Method** | **Description** |
| boolean endsWith(String suffix) boolean startsWith(String prefix) | Returns true if this string ends with or begins with the substring specified as an argument to the method. |
| boolean startsWith(String prefix, int offset) | Considers the string beginning at the index offset, and returns true if it begins with the substring specified as an argument. |
| int compareTo(String anotherString) | Compares two strings lexicographically. Returns an integer indicating whether this string is greater than (result is > 0), equal to (result is = 0), or less than (result is < 0) the argument. |
| int compareToIgnoreCase(String str) | Compares two strings lexicographically, ignoring differences in case. Returns an integer indicating whether this string is greater than (result is > 0), equal to (result is = 0), or less than (result is < 0) the argument. |
| boolean equals(Object anObject) | Returns true if and only if the argument is a String object that represents the same sequence of characters as this object. |
| boolean equalsIgnoreCase(String anotherString) | Returns true if and only if the argument is a String object that represents the same sequence of characters as this object, ignoring differences in case. |
| boolean regionMatches(int toffset, String other, int ooffset, int len) | Tests whether the specified region of this string matches the specified region of the String argument.  Region is of length len and begins at the index toffset for this string and ooffsetfor the other string. |
| boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len) | Tests whether the specified region of this string matches the specified region of the String argument.  Region is of length len and begins at the index toffset for this string and ooffsetfor the other string.  The boolean argument indicates whether case should be ignored; if true, case is ignored when comparing characters. |
| boolean matches(String regex) | Tests whether this string matches the specified regular expression. Regular expressions are discussed in the lesson titled "Regular Expressions." |

* **The String Builder Class**
  + StringBuilder objects are like String objects, except that they can be modified internally.
  + These objects are treated like variable length arrays that can contain a sequence of characters.
  + At any point length and content of the sequence can be changed through method invocations.
  + **Length and Capacity**
    - The StringBuilder class, like String class has a length() method that returns the length of the character sequence in the builder.
    - Unlike Strings, every string builder has a capacity, the number of character space that has been allocated.
    - The capacity which is returned by the capacity() method.
    - The capacity is always greater than or equal to length and will automatically expand as necessary to accommodate additions to the string.

|  |  |
| --- | --- |
| **StringBuilder Constructors** | |
| **Constructor** | **Description** |
| StringBuilder() | Creates an empty string builder with a capacity of 16 (16 empty elements). |
| StringBuilder(CharSequence cs) | Constructs a string builder containing the same characters as the specified CharSequence, plus an extra 16 empty elements trailing the CharSequence. |
| StringBuilder(int initCapacity) | Creates an empty string builder with the specified initial capacity. |
| StringBuilder(String s) | Creates a string builder whose value is initialized by the specified string, plus an extra 16 empty elements trailing the string. |
| **Length and Capacity Methods** | |
| **Method** | **Description** |
| void setLength(int newLength) | Sets the length of the character sequence. If newLength is less than length(), the last characters in the character sequence are truncated. If newLength is greater than length(), null characters are added at the end of the character sequence. |
| void ensureCapacity(int minCapacity) | Ensures that the capacity is at least equal to the specified minimum. |

* + **String Builder Operations**

|  |  |
| --- | --- |
| **Various StringBuilder Methods** | |
| **Method** | **Description** |
| StringBuilder append(boolean b) StringBuilder append(char c) StringBuilder append(char[] str) StringBuilder append(char[] str, int offset, int len) StringBuilder append(double d) StringBuilder append(float f) StringBuilder append(int i) StringBuilder append(long lng) StringBuilder append(Object obj) StringBuilder append(String s) | Appends the argument to this string builder. The data is converted to a string before the append operation takes place. |
| StringBuilder delete(int start, int end) StringBuilder deleteCharAt(int index) | The first method deletes the subsequence from start to end-1 (inclusive) in the StringBuilder's char sequence. The second method deletes the character located at index. |
| StringBuilder insert(int offset, boolean b) StringBuilder insert(int offset, char c) StringBuilder insert(int offset, char[] str) StringBuilder insert(int index, char[] str, int offset, int len) StringBuilder insert(int offset, double d) StringBuilder insert(int offset, float f) StringBuilder insert(int offset, int i) StringBuilder insert(int offset, long lng) StringBuilder insert(int offset, Object obj) StringBuilder insert(int offset, String s) | Inserts the second argument into the string builder. The first integer argument indicates the index before which the data is to be inserted. The data is converted to a string before the insert operation takes place. |
| StringBuilder replace(int start, int end, String s) void setCharAt(int index, char c) | Replaces the specified character(s) in this string builder. |
| StringBuilder reverse() | Reverses the sequence of characters in this string builder. |
| String toString() | Returns a string that contains the character sequence in the builder. |

* **Autoboxing and Unboxing**
  + Autoboxing is an automatic conversion that the java compiler makes between the primitive types and their corresponding object wrapper classes.
  + For example converting int to Integer, a double to Double and so on.
  + If the conversion goes other way this is called unboxing.

List<Integer> li = new ArrayList<>();

for (int i = 1; i < 50; i += 2)

li.add(i);

* + Although you add the int values as primitive types rather than Interger object to li, the code compiles.
  + Because li is a list if Integer objects not a list of int values.
  + Thus the compiler converts the previous code to the following at run time.

List<Integer> li = new ArrayList<>();

for (int i = 1; i < 50; i += 2)

li.add(new Integer.valueOf(i));

|  |  |
| --- | --- |
| **Primitive type** | **Wrapper class** |
| Boolean | Boolean |
| Byte | Byte |
| Char | Character |
| Float | Float |
| Int | Integer |
| Long | Long |
| Short | Short |