Cheat Sheet #2

Constants, Variables, Parameters, Scoping

Constants

Constants are declared **static final** to make them accessible without having to have an instance of that object and to make it immutable. The name is capitalized and words separated by underscores by convention.
public static String HELLO_WORLD = "Hello World!";

Variables

Variables have to be **declared** and **initialized** before usage. Each variable has a **type** and a **name**. If there is no explicit initialization, primitive variables get initialized with default values (0 or *false*), otherwise with *null*. One can declare and initialize variables in one step:
int i = 12;

Parameters

Each method takes 0 to n parameters. Those parameters are objects passed from outside of the method on the call. Those objects can be used inside of the method.
Primitive types are mostly passed **by value**, more complex types are usually passed **by reference**. This means changing values and fields of the reference passed changes those values outside of the method for this object as well. The reference actually is the representation of an object in storage.

Scoping

Each object only exist within a specified scope. A variable that has been declared inside a method (**local variable**) is only known inside of this method and cannot be accessed by methods outside of this method.
A variable declared in the body of a class (**class variable**) is known to all the class' methods. It is unique for each object created of this class.
If the variable is static, it only exists once for a class, but it can be accessed without having to instantiate an object of this class. It is known everywhere where the class itself is known.

Primitive Data Types

There are eight primitive data types in java:

- **byte** - 8-bit signed two's complement integer, [-128,127].
- **short** - 16-bit signed two's complement integer, [-32768,32767].
- int 32-bit signed two's complement integer, $[-2^{31}, 2^{31} - 1]$.
- **long** - 64-bit two's complement integer, $[-2^{63}, 2^{63} - 1]$.
- float - 32-bit IEEE 754 floating point
- double - 64-bit IEEE 754 floating point.
- char - single 16-bit Unicode character, $[\backslash u0000, \backslash ufffff]$.
- boolean - true, false

String is technically no primitive data structure.

Arithmetic expressions, math

There are some basic arithmetic operations available for numerical types in java. Those include:

- addition (a + b)
- subtraction (a - b)
- multiplication (a * b)
- division (a / b)
- modulo/remainder (a % b)
- pre/post increment (++a / a++)
- pre/post decrement (–a / a–)

For more advanced operations the **Math**-library provides many operations like computing the square root, the power, sin and cos of numbers and so on.

Type Conversion

To convert one type into another one there is the possibility of casting. This will work only if the types are compatible. One may loose some precision for numerical types.
int i = (int)5.5;
For known types java does the type conversion implicitly, e.g. printing the value of an integer.
int i = 7; System.out.println("i:" + i);
There are some methods in the Wrapper classes of the primitive types that provide conversion from other primitive types, like Integer.parseInt(...) to convert from String to int.

Logical Expressions

- **AND** - a & b or a && b (lazy evaluation)
- **OR** - a | b or a || b (lazy evaluation)
- **XOR** - a ^ b
- **NOT** - !a

Operators, operator precedence

Besides the arithmetic operators, there are also logical bitwise operators (& - AND, | - OR, ^ - XOR, ~ - (Inversion)), operators to assign values (& =, |=, ^=, <<=, >>=, >>>=) and also the shift operators (<<, >>, >>>).
Precedence From highest to lowest:
[],(),.,++,–,+ (unary), -(unary),~, !, (type), new, *, /, %, &, +, -, + (string concatenation), <<, >>, >>>, <, <=, >, >=, instanceof, ==, !=, &&, ^, |, , ||

Characters and Strings

Basically **char** is a primitive type referring to one single character. **String** is a class referring to 0 to n characters.
A string is **immutable**, so every time a new value is assigned to a String, a new String object is created and stored.

String expressions

There are many useful functions for the handling of Strings in JAVA, most of them located in the **String** class.
To concatenate Strings: *StringA + StringB;*
To get length of String: *StringA.lengt();*
To check if a String contains another string: *StringA.contains(StringB);*
To use variable values inside of a String: *String.format("Some integer: %d, some String: %s, some floeat: %.2f", integer, string, floatingNumber);*

Basic I/O

For basic input one can use the **Scanner** class of java.
Scanner scanner = new Scanner(System.in);
scanner.nextInt();
For basic output one can use the **System.out** class:
System.out.println("Hello World");

Exceptions and exception handling

In JAVA code may throw exceptions. These exceptions will stop the execution of the code if not handled. To handle exceptions one can either declare the method to **throw** this exception - the exception will then have to be handled elsewhere - or use a **try-catch**-block to define alternative behavior.
There are **checked** and **unchecked** exceptions in JAVA. Unchecked exceptions are not checked at compile-time, only on runtime. Checked exceptions will already be checked during compilation.

Sources

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html
http://en.wikibooks.org/wiki/Java_Programming/Arithmetic_expressions
http://bmanolov.free.fr/javaoperators.php
http://beginnersbook.com/2013/04/java-checked-unchecked-exceptions-with-examples/