Java Cheat Sheet

Hello World

```
public class HelloWorld {
  public static void main (String [] args)
  {
    System.out.println("Hello World");
  }
}
```

Java Basics

`java edu.simpson.ClassName`	Run `main` in ClassName
`javac *.java`	Compile .java to .class

A class goes in a .java file.
Methods and attributes go in a class.
Statements go in methods.
A block is contained in { }

Comments

`//`	Comment to end of line
`/* x */`	Comment everything between
`/** x */`	Javadoc comment

Primitive Variables

type	bytes	range
`byte`	8	-128..127
`short`	16	-32,768..32,767
`int`	32	-2,147,483,648..2,147,483,647
`long`	64	-2^{63} to $2^{63}-1$
`float`	32	1.4e-45..3.4e+38
`double`	64	4.9e-324..1.7e+308
`char`	2	Unicode letter, 0..65,535
`boolean`		true..false

Variables should begin with a lower case letter. By default, numbers are `int` or `double`. Append F for float, L for long, and D for double. Use single quotes for a `char`.
Declaration:
`datatype variablename;`

Objects

Objects are created from classes.

```
Person myPerson;
myPerson = new Person();
```

Reference Variables

These contain a memory address where an object exists.
Access object variables with dot operator:
`myPerson.name="Fred";`
Access object methods with dot oeprator:
`int x = myPerson.getAge()`

Expressions

`=`	Assignment (Don't confuse with ==)
`*`	Multiply
`\`	Divide
`%`	Modulus (remainder)
`x++`	Return x, then increment
`++x`	Increment, then return x
`x+=2`	Add 2 to x, store in x
`x*=2`	Multiply x by 2, store in x
`func(x)`	Run code in `func`, return result

Conditionals

`==`	Equals
`<=`	Less than or equal
`<`	Less than
`>=`	Greater than or equal
`>`	Greater than
`!=`	Not equal
`!`	Not
`&&`	And
`||`	Or

Strings

`s1.equals(s2)`	Compare two strings
`s1.equalsIgnoreCase(s2)`	Compare, ignoring case
`s1.length()`	Return length of string
`String[]a=s1.split(" ");`	Split string separated by spaces

Loops

`while(i<10) { ... }`	Pretest
`do { ... } while (i<10);`	Post-test
`for(int i=0;i<10;i++) { ... }`	For loop

Branches

```
if(i<10){
  // do something
} else if(i>10) {
  // do something else
} else {
  // do if nothing else matched
}
```

Classes

Classes contain a blueprint of all the attributes, and methods for an object.

```
public class Person {
  // Attribute
  private int age;
  private String name;
  // Constructor that sets the name
  public Person(String name) {
    this.name=name;
  }
  // Method that returns an age
  public int getAge() {
    return age;
  }
  // Method that sets an age
  public void setAge(int age) {
    this.age = age;
  }
}
```

Methods

Simple method:

```
public void doSomething() { ... }
```

Method that returns a value:

```
public int getIntegerNumber() { ... }
```

Method with two parameters:

```
public void setSize(int height, int width) { ... }
```

Static

Static methods are called without creating an object.
Static methods may not access non-static variables or methods.
Static variables are shared across all instances.

Inheritance

A subclass extends a superclass.
A child extends a parent.
A child inherits all the parent's attributes and methods.
Methods can be overridden with new functionality, attributes can not.
All objects have the `Object` class as their top parent.
To create a child: `class Child extends Parent {`
To call a parent constructor (must be first line in the constructor): `super( ... );`

Interface

An interface is a pure abstract class that defines a protocol
To declare an interface:

```
public interface MyInterface {
  void myFunction(); // No method body
}
```

Input

Get input from a user:

```
Scanner scan = new Scanner(System.in);
int a = scan.nextInt(); // Get an integer
String b = scan.next(); // Get text
```

Get input from a file:

```
FileInputStream in = new FileInputStream("file.txt");
Scanner scan = new Scanner(in);
```

Arrays

Create an integer array: `int [] a=new int[50];`
Assign first value: `a[0]=5;`
Assign last value: `a[49]=5;`

Libraries and packages

Import a package: `import java.util.Date;`

`$Revision: 1.0 $, $Date: 2007/11/27 $.`