Java Cheat Sheet

Hello World

By now, you should know this by heart! Eclipse has a kind soul and remembers the first line for you when you start a new class, but the rest is all on you.

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

Comments

Comments are a useful way to document your program. They are ignored by the compiler. You can type in anything you want like a description of the segment of code you are writing. This is also useful for other programmers that may be using or reading your code.

A comment starts with two forward slashes //. Anything that comes after the // up to the end of the line is a comment. Another form of the comment is the / */ pairing, where everything inside the start /* and end */ becomes a comment.*

```
public class CommentsTest {
	public static void main(String[] args) {
		System.out.println("This program is a demonstration of comments.");

		// Comments are lines that are not looked at by the compiler (Eclipse)

		// This is a comment that has
		// been put on more than
		// one line

		/*	This is a comment that has
			been put on more than
			one line
		*/

		// System.out.println("Comments work great for ignoring lines of");
		// System.out.println("code that you're not sure you need…");
	}
}
```

Variables & Math

Just as in algebra, variables store values. Variable names can be any combination of letters, numbers and an underscore (_) as long as it starts with a letter or an underscore. Associated with all variables is a data type (int, double, bool, char, String). A variable's data type can never change once it has been declared.

```
public class SimpleCalculator {
	public static void main(String[] args) {
		int x = 5;
		int y = 7;
		int z = x + y;		// ints (short for integers) are numbers without
					// decimals

		double a = 9.4;
		double b = 2.333333333;
		double c = a / b;		// doubles are numbers with decimals

		bool winning = true;		// bools (short for Boolean) are variables
						// that can only be true or false

		char myFavoriteLetter = 'j';  // chars are surrounded by 'single quotes'

		String myName = "Scrabble";		// Strings (capital S) are groups of
							// chars. They are surrounded
							// by "double quotes".
	}
}
```

Getting User Input

In Java, we need to import a built-in class called Scanner. We do this on the very first line of your program. Then, in your "main" method, we create a new Scanner, and use it (it's called "s" in this example) to get a nextInt() or a nextLine(), etc. and store the user input into a variable with an appropriate data type.

```
import java.util.Scanner;

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

		System.out.println("What is your name?");
		String name = s.nextLine();

		System.out.println("Nice to meet you, " + name);
	}
}
```

If Statements

Less than	*<*
Greater than	*>*
Less than or equal to	*<=*
Greater than or equal to	*>=*
Equal to	*== (i.e. 2 equal signs)*
Not equal to	*!=*

Be careful not to confuse the assignment operator (=) with the equal to operator (==).

```
4 > 3;   // this expression is true
9 != 0;  // this expression is true
3 == 9; // this expression is false
```

&& (and) *and* || (or) *are used to compare more than one expression at the same time.*

In && (and)*, both expressions have to be true for the entire expression to be true.*
In || (or)*, at least one expression has to be true for the entire expression to be true.*

When an expression is not true, it is false.

Whenever you need your program to make a decision between two or more possible actions, you can use an if statement. Note that there is no semicolon; after the if() line. What this if statement means is that if the expression within the parentheses is true, then go and execute the lines within the body (between the { }'s). Otherwise, if the expression is false, do not go there. You can enter as many statements/expressions you want within the { }'s using && and/or ||.

```
import java.util.Scanner;

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

		System.out.println("What is your age?");
		int age = s.nextInt();

		if(age > 70) {
			System.out.println("Enjoy retirement!");
		} else if(age > 50) {
			System.out.println("You must still enjoy your job…");
```

```
            } else if(age > 30) {
                System.out.println("How are the children?");
            } else if(age > 18) {
                System.out.println("How's college?");
            } else if(age > 10) {
                System.out.println("Enjoy childhood while you can.");
            } else if(age < 10 && age > 0) {
                System.out.println("You're too young for a computer…");
            } else {
                System.out.println("Liar…");
            }
        }
}
```

While Loops

A condition can be a comparison or any kind of Boolean value. Before, looping the code in the body of the while, the program checks to see if the condition inside the parentheses is true. If it is, the body will start executing, and keep executing until the condition becomes false.

```
public class MyGame {
    public static void main(String[] args) {
        int numberOfLives = 5;

        while(numberOfLives > 0) {          // makes sense, right?
            // put your game code here….

            if(dead == true) {
                numberOfLives--;    // a quick way to subtract one
                                    // from numberOfLives.
            }
        }
    }
}
```

For Loops

A for loop is a similar to a while loop, but still depends on a Boolean condition looping. You can initialize a value for counting, then compare it against another value and change the value (known as incrementing).

```
public class ForTest {
    public static void main(String[] args) {
        for(int i = 0; i < 4; i++) {
            System.out.println("The number is " + i);
        }
    }
}
```

Notice that this for loop does the same thing as the while loop example above. This is a list of steps that a for loop goes through:

1. initialize the counter variable (i)
2. check if condition is true
    if false, loop stops
    if true, go to step 3
3. execute lines in the body
4. increment/decrement the value
5. go to step 2

Random Numbers

Just like we imported the Scanner class, now we need to import the Random class to generate random numbers. Once we import the Random class and create a Random object called "r", we can have "r" give us a nextInt(), nextDouble(), etc., and manipulate those numbers till we get the range of our choosing.

```
import java.util.Random;

public class RandomTest {
	public static void main(String[] args) {
		Random r = new Random();

		// print 10 random integers
		for(int i = 0; i < 10; i++) {
			System.out.println(r.nextInt());
		}

		// print 10 random integers between 0 and 5
		for(int i = 0; i < 10; i++) {
			System.out.println(r.nextInt(5));
		}

		// print 10 random integers between 20 and 30
		for(int i = 0; i < 10; i++) {
			System.out.println((20+r.nextInt(10)));
		}
	}
}
```

Arrays

Arrays are a great tool in any language to store multiple items inside of one variable. Think of an array as a group of containers. For example, if you have an integer array of size 10, picture it as 10 containers ready to hold one integer each. In Java, we need to both declare and initialize arrays so the containers are ready for use.

Hints and mind-blowers:

- *Array "indexes" start at 0 and end at the size of your array minus one. For example, if you have an array of size 5, your indexes would be 0, 1, 2, 3, and 4.*
- *Strings are really arrays of characters...deep, right?*
- *For loops and arrays are a match made in heaven…try to figure out why!*

```
public class ArrayTest {
	public static void main(String[] args) {
		char[] favoriteSymbols = new char[4];

		favoriteSymbols[0] = '*';
		favoriteSymbols[1] = 'W';
		favoriteSymbols[2] = '7';
		favoriteSymbols[3] = '%';
	}
}
```

Methods

In Java, methods are things that help avoid having to duplicate code/logic in your program. If you find yourself doing something more than once, chances are that it should be made into a method. Methods are created once and can be "called" (aka ran) multiple times from almost anywhere in your code. In fact, the "public static void main" line you should have memorized right now is it's own method declaration.

A method declaration includes five main parts:

- *Public vs. private (for now, we'll stick with public)*
- *Static vs. non-static (for now, we'll stick with static)*
- *The return type – if your method returns an integer, type int here. If it returns nothing, like the main method, type void.*
- *The name of the method*
- *The parameters (inside the parentheses) – these are variables that you can "pass into" the method and use inside*

```
public class MethodToMyMadness {
	public static void main(String[] args) {
		int mySum = addTwoNumbers(5, 7);	// this should equal 12
		System.out.println(mySum);
	}

	public static int addTwoNumbers(int a, int b) {
		return a + b;
	}
}
```

Final Project Ideas

- Blackjack
- Hangman
- Tic-Tac-Toe
- “Create Your Own Adventure” Game
- Virtual Store
- Who Wants To Be A Millionaire?
- Minesweeper
- Encryption Tool (encode/decode secret messages)
- Horse Race Game

If you’re bored and/or “done,” ask me about…

- Classes
- File I/O
- Recursion
- ArrayLists
- Search algorithms