Java Basics Reference

Basic Template

In Java every program is also a class and in that class you will have a main method. All of your early programs will have the same basic setup:

```
class Basic1
{
     public static void main(String []args)
     {
          // Your code goes here
     }
}
```

- replace **Basic1** with whatever name you want but you must save the file with the same name you choose plus a ".java" extension.
- In Java methods are either **public** or private. If they are private they can not be accessed from outside of the class. main needs to be public or it can not be called.
- The **static** modifier means that the method belongs to the class, and as such we don't need an instance of the class (an object) to call the method.
- **void** describes what the main method returns. methods can return any type of value, but if they don't return anything at all the return type is void.
- The main method must be called **main**.
- `String []args` – it is possible to pass in parameters to your program when you call it. These parameters are always passed in as an array of Strings when they are supplied. The name "args" refers to arguments but you can use any parameter name you like.

Basic I/O

Output

There are three commands we use to output to the screen in Java.

`System.out.println()`	- prints a single value then a newline
`System.out.print()`	- prints a single value
`System.out.printf()`	- prints a formatted string that can contain multiple values. The values in variables get substituted into the string.

int age = 15;
double cost = 12.345;
String name = "Joe";

%d	– int
%f	– float/double
%s	– String
-	– left align

Statement	**Output** (~ represents formatted spaces)
`System.out.println("Hi " + name + " I hear you're " + age);`	`Hi Joe I hear you're 15`
`System.out.printf("%10d", age);`	`~~~~~~~~15`
`System.out.printf("%s, you owe %8.2f", name, cost);`	`Joe, you owe ~~~12.35`
`System.out.printf("Bill, with tip $%.2f", cost*1.15);`	`Bill, with tip $14.20`

Input - Scanner (import java.util.)*

The Scanner class can do a lot of cool things with regular expressions but as far as we are concerned it just makes console input simple.

```
import java.util.*;
public class Input {
       public static void main (String [] args)
       {
              Scanner stdin = new Scanner(System.in);        // only need to do this once

              int num = stdin.nextInt();
              String name = stdin.next();
              BigInteger bigNum = stdin.nextBigInteger();
              double cost = stdin.nextDouble();
              if(stdin.hasNextInt())
                     num = stdin.nextInt();                  // some basic input checking
       }
}
```

Variables and Types

Primitive type	Size	Examples	Wrapper type	Note
boolean	-	true, false	**Boolean**	
byte	8-bit	'X', 'a', ';'	**Byte**	stored as ASCII value
char	16-bit	'A', 'ي', 'Ψ'	**Character**	Unicode
int	32-bit	12, -4	**Integer**	
double	64-bit	3.4, -8.65	**Double**	

Assignment and Boolean Operations

Use this	To do this	Example
=	Assign a variable a new value	age = age + 1
==	check equality	if (x == 20)
>	check for Greater than	apples > oranges
<	check for Less than	speed < 100
&&	check for both conditions (AND)	(age >= 19) && (country.equals("Canada"))
| |	check for either condition (OR)	DoneNow | | (x > 10)
!	check for the opposite condition	! (age > 17 && money > 20)
!=	not equals	x != 0

Mathematical Operators

Use this	To do this	Example	
+ - *	add, subtract, multiply two numbers	a = x + 3	

/	divides two numbers, promotes answer to the most precise used.
	int / int → int
	float / int → float
	double / int → double
		a = 7/3
		a = 7.0 / 2
			// (a = 2)
			// (a = 3.5)

%	Remainder of first/second (Mod)	a=7/3	// (a = 1)
?:	takes on the one of two values based on the condition. **(***cond* **?** *trueVal* **:** *falseVal* **)**	a = (5>2 ? 12 : 20);	// (a=12)
++	increment by one	a++	
--	decrement by one	a--	

If Statements

If statements in Java are the same as Turing, except the syntax has changed a bit. Just like in Turing, you do not need to use an else branch, or an else if branch.

```
int testscore = 76;
char grade;

if (testscore >= 80) {
    grade = 'A';
} else if (testscore >= 70) {
    grade = 'B';
} else if (testscore >= 60) {
    grade = 'C';
} else if (testscore >= 50) {
    grade = 'D';
} else {
    grade = 'F';
}
System.out.println("Grade = " + grade);
```

Boolean Operators

`|`	the OR operator
`&`	the AND operator
`^`	the XOR operator
`!`	the NOT operator
`||`	the short-circuit OR operator
`&&`	the short-circuit AND operator
`==`	the EQUAL TO operator
`!=`	the NOT EQUAL TO operator

short-circuit means that if the expression can be evaluated without checking the second value then don't check the second value. e.g.

```
        if( x != 0 && 100/x > 10)
```

if x=0 then the first term is false, and because false and anything is false the whole thing is false, thus preventing a division by zero.

Conditional Expression

The conditional expression is an expression that takes on a value based on a condition. Although many people avoid it because of its cryptic nature it can make some code much shorter. The syntax is:

<condition> ? <true value> : <false value>

These two boxes do the same thing:

```
if(x>10){
      x = 10;
}
else{
      x = y;
}
```

```
x = y > 10 ? 10 : y;
```

Loops

There are three loops in Java, the while loop, the do…while loop and the for loop.

while loop

```
double mass = 10;
int count = 0;
while(mass > 1){         // continue to loop while mass is greater than one
      count += 1;
      mass /= 2;
      System.out.printf("Day:%3d Mass:%5.2f\n", count, mass);
}
```

do …while loop
The do…while loop is not used very often, but it is useful when you need to enter the loop at least once.

```
double mass = 10;
Scanner stdin = new Scanner(System.in);
int choice;
do{
      System.out.println("Enter a number(1-10)");
      choice = stdin.nextInt();
}while(choice < 1 || choice >10);
```

for loop

```
int []nums = {23,453,564,5,3,54,98,65,74,4};

for(int i=0; i<10; i++){            // i is local to the for loop
      nums[i] = nums[i] * 2;        // starts at 0, loops as long as i<10
}                                   // and increases by one after each loop

for(int i : nums){                  // i takes on the value of each element
      System.out.println(i);        // in the array nums
}
```

break
Any loop can be broken out of prematurely with a break. Using break is considered poor style, so try to avoid using it.

```
Scanner stdin = new Scanner(System.in);
String pass;
while(true){
      System.out.println("Enter the password:");
      pass = stdin.next();
      if(pass.equals("Massey")){
            break;
      }
      System.out.println("Invalid password");
}
System.out.println("Access granted");
```