Simple Java Program

```
public class HelloWorld{
    //Main method must take String[] args
    public static void main(String[] args){
        //Main functionality here
    }
}
```

Parsing

int *name = Interger.parseInteger(str)

double *name = Double.parseDouble(str)

Casting: (int)(*some other type); (double)* (some other type)

Java Object

```
public class ObjectName{
    //variables that are private to this object
    private type variableName;

    //Constructor gets called when new method is created --> Multiple constructors
    //can exist
    public ObjectName(*parameters){
        //Whatever happens in constructor
    }
}
```

Math Methods

Math.pow(a, b)	Math.PI()
Math.log(x), Math.log10(x)	Math.sqrt(x)
Math.floor rounds down	Math.ceil() rounds up
Math.random() Unif[0,1)	Math.min(), Math.max()

Uniform Random Int between [1,6] --> (int) (Math.random() + 1)

IntegerStack

```
IntegerStack intStack = new IntegerStack();
//Add an element to the stack
intStack.push(int);
//Removing the top element from the stack
intStack.pop();
//checking if empty
intStack.isEmpty();
```

Might need to import IntegerStack --> Class specific object --> not java util

Knapsack

```
P = integer array (n+1, V+1)
for(v = 0 ... V)
P(0,v) = 0;
for (i = 1 ... n)
for (v = 0 ... V)
if (volumes(i-1) <= v)
P(i,v) = max(profit(i-1) + P(i-1,v-volumes(i-1)),
P(i-1,v));
else
P(i,v) = P(i-1,v)
return P(n,V);
```

Statements

If Statement

```
if ( expression ) {
  statements
} else if ( expression ) {
  statements
} else {
  statements
}
```

While Loop

```
while ( expression ) {
  statements
}
```

Do-While Loop

```
do {
  statements
} while ( expression );
```

For Loop

```
for ( int i = 0; i < max; ++i) {
  statements
}
```

For Each Loop

```
for ( var : collection ) {
  statements
}
```

Switch Statement

```
switch ( expression ) {
  case value:
    statements
    break;
  case value2:
    statements
    break;
  default:
    statements
}
```

Exception Handling

```
try {
  statements;
} catch (ExceptionType e1) {
  statements;
} catch (Exception e2) {
  catch-all statements;
} finally {
  statements;
}
```

for loop is more general: for(int i; booleanMethod(i), incrementMethod(i){}
incrementing in short:
i = i + 1; --> i++;
i = i - 1; --> i--;
i += a;
i -+ a;

By **curae**
cheatography.com/curae/

Published 9th December, 2018.
Last updated 9th December, 2018.

Insertion Sort

```
Insertion Sort:
for (i = 1 ... n-1)
for (j = i ... 1)
if (a(j-1) > a(j))
swap(a(j-1),a(j))
else break;
```

String Methods

.toUpperCase();	toLowerCase();
.substring(i,j) j is excluded	.length()
.compareTo(str) *lexicographic ordering (-1, 0, 1)	.equals(str)
.indexOf(e)	.concat(str)
.charAt(i)	.contains(e)

Arrays

```
type[] arrayName = new type[length]
E.g.
boolean[] visitedNode = new
boolean[this.numberOfNodes];
```

ArrayList

create	ArrayList<type> name = new ArrayList<type>();
access element	list.get(i)
update element	list.set(i, e)
return length	list.size()
add element *somewhere*	list.add(e)
add element at i	list.add(i,e)
remove element	list.remove(i or e)

ArrayList (cont)

remove all elements	list.clear()

```
import java.util.ArrayList;
```

Queues

```
Queue<type> q = new Queue<type>();
//put element in queue
q.enqueue(e);
//remove element in queue
q.dequeue();
//check if empty
q.isEmpty();
//check size
q.length(); or q.size();
```

We use a class specific Queue method --> not the java utils one

GCD

```
public static int GCD (int m, int
n){
    int temp;
    while (n%m != 0){
      temp = m;
      m = n%m;
      n = temp;
      if (m==0) return 1;
    }
    return m;
  }
```