

Simple Java Program

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

Parsing

```
int *name = Integer.parseInt(str)
double *name = Double.parseDouble(str)

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

Java Object

```
public class ObjectName{
    //v ari ables that are
    private to this object
    private type variab -
    leName;

    //C ons tructor gets
    called when new method is
    created --> Multiple constr -
    uctors
    //can exist
    public Object Nam e(* -
    par ame ters){
        //W hatever
        happens in constr uctor
    }
}
```

Math Methods

Math.pow(a, b)	Math.PI()
Math.log(x), Math.log-10(x)	Math.sqrt(x)
Math.floor rounds down	Math.ceil() rounds up

Math Methods (cont)

Math.random()	Math.min(),
Uniff(0,1)	Math.max()

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

IntegerStack

```
IntegerStack intStack = new
IntegerStack();
//Add an element to the stack
intSta ck.p us h(int);
//Removing the top element from
the stack
intSta ck.p op();
//checking if empty
intSta ck.i sE mpty();
```

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 (volum es(i-1) <= v)
P(i,v) = max(pr ofi t(i-1) +
P(i-1, v-v olu mes (i-1)),
P(i-1,v));
else
P(i,v) = P(i-1,v)
return P(n,V);
```

Statements

If Statement

```
if ( expre ssion ) {
    st at e ments
} else if ( expre ssion ) {
    st at e ments
} else {
    st at e ments
}
```

While Loop

```
while ( expre ssion ) {
    st at e ments
}
```

Do-While Loop

```
do {
    st at e ments
} while ( expre ssion );
```

For Loop

```
for ( int i = 0; i < max; ++i) {
    st at e ments
}
```

For Each Loop

```
for ( var : colle ction ) {
    st at e ments
}
```

Switch Statement

```
switch ( expre ssion ) {
    case value:
        st at e ments
        break;
    case value2:
        st at e ments
        break;
```

Statements (cont)

```
> 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;
```

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)

String Methods (cont)

.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)
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;
}
```



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

Published 9th December, 2018.
 Last updated 9th December, 2018.
 Page 2 of 2.

Sponsored by **Readable.com**
 Measure your website readability!
<https://readable.com>