

Simple Java Program

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

Math.log10(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
```

Statements (cont)

```
} 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;

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

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

Arrays

```

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

ArrayList

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

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;
}
    
```

