Type casting:

Type cast can be used to change the data type of a value from it's declared → another data type

```
Variable = ([DataType]) value;
```

Can be change object to specific object type EG researchStudent or CourseWorkStudent

<u>Instantiate class</u> → Object:

```
[ClassName] [Class variable] = new [ClassName]();
```

<u>Create array of objects:</u> (Inside array is nulls. Think of it as creating an array of numbers)

```
[ClassName][] [Class variable] = new [ClassName][15];
```

<u>Instantiate the new objects for array (set objects) TIP: loop:</u> (WHY? Because w/o array is null)

```
[Class variable][i] = new [ClassName]();
```

Pass array → Method:

```
(person)
...([ClassName][] person)) ...
```

Sequential search (searching an array):

```
Boolean found = false;
if (studentList.length > 0) {
   int i= 0;
   while (!found) && (i < studentList.legnth)) {
      currentStudent = studentList[i];
      if(currentStudent == neededStudent) {
            found = true;
            i++;
            }
      }
}</pre>
```

<u>Create arrayList of objects:</u> (Inside array is nulls. Think of it as creating an array of numbers)

```
ArrayList<[ClassName]> arrayListName = new ArrayList<[ClassName]>();
```

Add the arrayList with new objects: (WHY? Because w/o array is null)

```
ClassName [Class Variable] = null; //Student student = null;
[Class Variable] = new [ClassName](); //Now we can use it inside if-statement
arrayListName.add([Class Variable]);
```

Searching through an arrayList:

```
for ([ClassName] [objectName] : [arrayListName]) {
    objectName.Method();
}
```

Remove a studentName from an arrayList of students:

```
For (Student person: [arrayListName]) {
    String Name = Person.GetName();

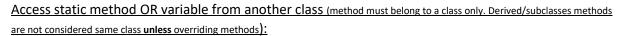
    if (Name.equals("Bob") {
        arrayListName.remove(person);
    }
}
```

Access an element of arrayList

```
[ClassName] [objectName] = [arrayListName].get(i);
[objectName].Method();
```

Pass arrayList → Method:

```
(studentList)
...(ArrayList<[ClassName]> arrayListName)...
```



```
[ClassName].Method();
[ClassName].Variable
```

Access static method from same class (derived/sub classes are considered same classes of parent):

```
Method();
... = Variable;
```

Creating a base class:

```
public class className1 {
     private int variable1;
     private String variable2;
     public ClassName1() { //Default constructor
          variable1 = 0;
          variable2 = "None";
     Public ClassName1(int initial.., String intialVaria...) {
          variable1 = initialVariable1;
          variable2 = initialVariable2;
     }
     Public void SetVariable1(int newVariable1) {
          variable1 = newVariable1;
     }
     Public int GetVariable1() {
          Return variable1;
     }
     Public void WriteOutput() { //Overriding method
           System.out.println("variable1: " + variable1);
          System.out.println("variable2: " + variable2);
```

```
public class ClassName2 extends ClassName1 {
       private int variable3;
       public ClassName2() { //Default constructor
              super(); //Calls OR Inherits className1 Default constructor
              variable3 = 0;
       Public ClassName2(int initial.., String intialVaria..., int in.)
              super(initialVariable1, intialVariable2); //Call OR Inh. className1 constructor
              variable3 = initialVariable3;
       Public void SetVariable3(int newVariable3) {
              variable3 = newVariable3;
       Public int GetVariable3() {
              Return variable3;
       Public void WriteOutput() { //Overriding method
              System.out.println("variable1: " + GetVariable1()); //calls oR Inh. className1 method
              System.out.println("variable2: " + GetVariable2()); //Calls OR Inh. className1 method
              OR
              super.WriteOutput();
              System.out.println("variable3: " + variable3);
       }
```

Creating a default constructor:

```
public ClassName() {
    variable1 = 0;
    variable2 = "None";
}
```

Creating a constructor:

```
Public ClassName(int initialVariable1, String intialVariable2) {
    variable1 = initialVariable1;
    variable2 = initialVariable2;
}
```

Writing txt file:

```
String filename = "out.txt";
PrintWriter = null;
Try {
    outputStream = new PrintWriter(filename);
    //Code IN here
    outputStream.write(string x)
} catch(FileNotFoundException e) {
    Output "Error can't write since not found";
    System.exit(0);
}
//Code IN HERE
outputStream.close();
```

Reading txt file:

```
Scanner [class variable] = null;
Try {
     [class variable] = new Scanner(new File("out.txt"));
} catch(FileNotFoundException e) {
     Output "Error opening file";
}
While([class variable].hasMoreTokens()) { [check if there is next..]
     String name = [class variable].next(); [move to next word and read]
     ...
     [class variable].nextToken(); [move to next line]
}
[class variable].close();
```

Writing binary file:

```
String filename = "out.dat";
Try {
ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(filename))
    outputStream.writeInt(name);
    outputStream.close();
} Catch(FileNotFoundException e) {
    Output "Problem opening file";
} Catch(IOException e) {
    Output "Problem writing to file";
}
```

Reading binary file: