These instructions are required for ALL the code you submit

Please follow them completely, or we will not be able to grade you code, and you will receive a 0 grade for the assignment.

- Use the submit (https://submit.cs.biu.ac.il/) system.
- All your code should reside in a single `.zip` file named `assN.zip` where `N` is the assignment number.
- When the zip file is extracted, it should create a sub-directory called `src` and the ANT config file `build.xml` (see next section). All your other code should reside under the `src` directory. You can create other directories if you want. The command line to correctly zip the assignment is `zip -r assN.zip build.xml src`.
- When compiling your code, we will export the compiled `.class` files into a sub-directory named `bin`. This is a common project structure.
- We may ask you to call certain classes in certain names. Please do so.

Checkstyle

- You must follow the [[CodingStyle]] and pass the checkstyle test with no errors.

User IDs

- On the main class `.java` file (or any file with a `main` function), add your ID in a comment on the head of the file.

ANT - A New Build Tool

In your "Introduction to Computing" course, you have learned about a simple utility named `makefile`. You have used a makefile to organize your compilation neatly, so that instead of typing a complex or cumbersome command every time you want to rebuild your project, you could just type `make compile` and the makefile knew what to do.

You should now understand that `makefile` is just one instance (in fact, a quite primitive one) of a class of software programs termed **build tools**. Build tools are programs that automate the creation of executable applications from source code. They are aiding software development by automating a wide variety of tasks that software developers do in their day-to-day activities, such as:

- Managing and downloading dependencies.
- Compiling source code into binary code.
- Packaging that binary code.
- Running tests.
- Deployment to production systems.

Throughout the assignments of this course, we will use Apache Ant (https://ant.apache.org/) as our build tool. Ant is a simple build tool; Nowadays, real-world Java projects use more powerful build tools such as Maven, SBT or Mill. However, it fits our purpose, and it does provide important utilities that will aid us to unify the testing of your assignment across platforms.

In Ant, the main configuration file is called `build.xml`, which is somewhat equivalent to the `makefile` you are already familiar with. As you can understand from its extension, it is an XML (https://en.wikipedia.org/wiki/XML) file, which specify a certain format for files so that machines can unambiguously retrieve information from it.
Similarly to `makefile`, the `build.xml` file specifies certain *targets* (i.e. automated tasks you'll want to use over and over again) and how to perform them. After writing a `build.xml` and putting it in your current directory, you can execute a target by running the command `ant <target-name>`. This will invoke the `Ant` software, which in turn will search for and read the `build.xml` and perform the the specified task. For more information about Ant, refer to online tutorials such as tutorialspoint (https://www.tutorialspoint.com/ant/index.htm).

Fortunately, we provide you in each assignment with a suitable build.xml file that will fit all your needs. Your requirements are:

1. [[Installing Ant]] on your personal computer (where you develop your code).
2. Download our provided `build.xml` for each assignment and put it under the root directory.
3. Before you submit, test your assignment by compiling and running it **using Ant**. It's crucial that you do so (and don't rely on the IDE `run` button for example) because our graders will use ant to compile and run you code - so this is your test.

To compile all your source files (which as mentioned, should reside in the `src` subdirectory), run: `ant compile`. To execute a program, use `ant run` when there is only one main class. In some of the assignments you have multiple tasks for which we instruct you to provide more than one main class (that is, a class with a `main` function you can run); in that case execute each task with `ant runI` (`I` being the task number).

In the `build.xml` we will provide for each assignment, you will also have:

- a `clean` target for cleaning up the binaries you have compiled. Please use it before you zip your assignment's root directory - you should not submit any `.class` files.
- a `check` target - this is for your convenience.

For Windows and Linux only: if you copy our provided `checkstyle-8.44-all.jar` and `biuoop.xml` to your root directory, `ant check` will run the checkstyle on all the `.java` files under `src` (and recursively its subdirectories, if any).
For Mac users, please run the checkstyle using the command line, as described in [[CodingStyle]]

You should download the `build.xml` file we will provide in the assignment, and include it without changes in the zip directory you submit. This is mainly for the convenience of grading your assignments. You are required to verify your assignment is properly compiling and running using ant, since this is how graders will check your work.

What we will run on your submission

We should be able to run the following commands (on unix) to run your code:

```
unzip ass1.zip
ant compile
ant run
```

To execute a `run` target with command line arguments (e.g. `10`, `arg2`, `3`), run:

```
ant -Dargs="10 arg2 3″ run
```

An example

Assuming we had assignment number 99, with two task: in task 1 your program was requested to print "Hello Task1", and in task 2 your program was requested to print "Hello Task2". Then the following file is a valid submission: ass99.zip (data/ass99.zip)