

Lab01: Getting started with Netbeans

Introduction to :

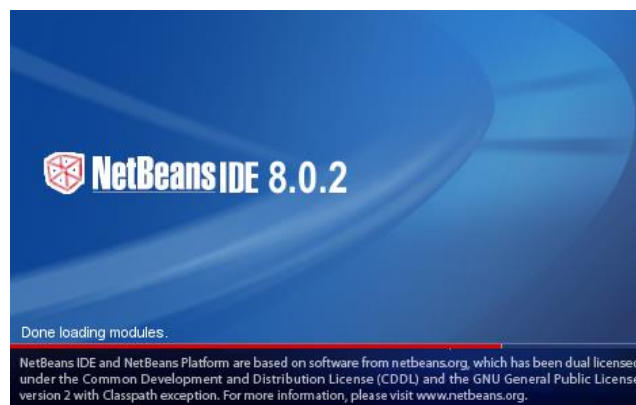
1. Netbeans IDE
2. Exercises

1. Netbeans IDE

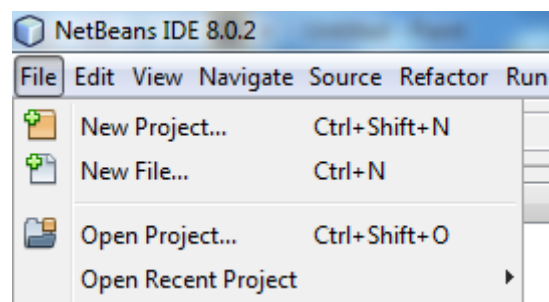
How to start and set up the project in Netbeans IDE?

Start NetBeans IDE:

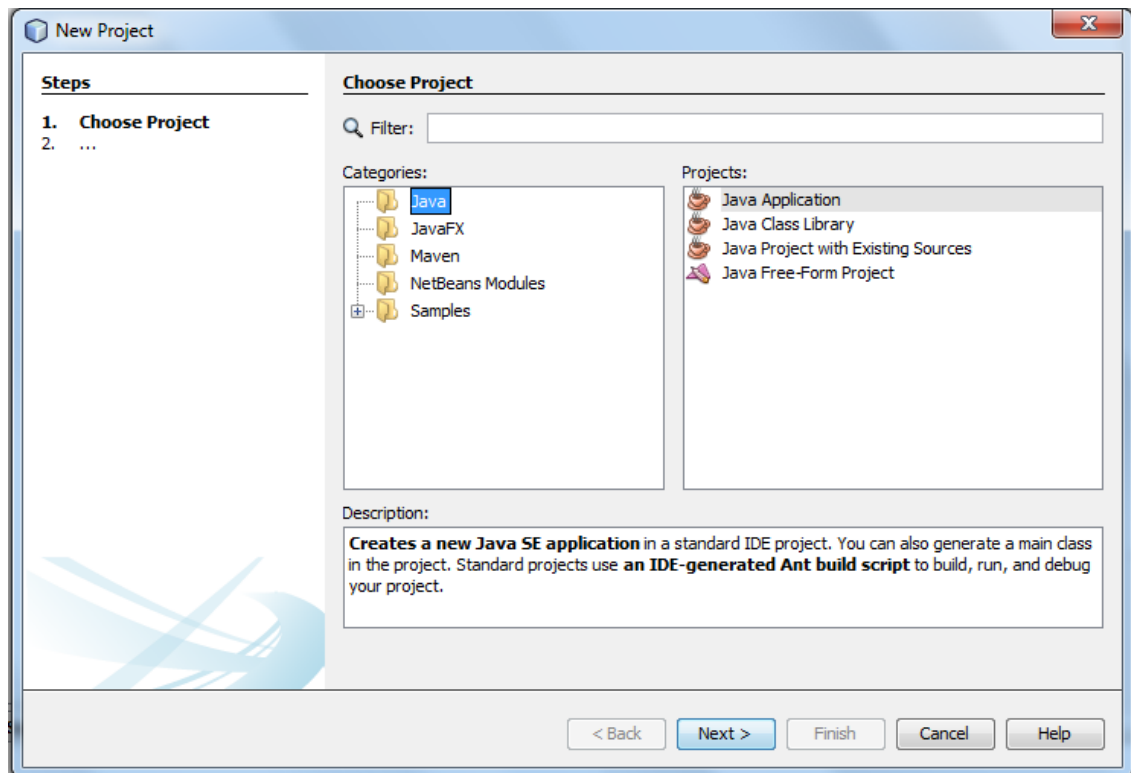
- Start Netbeans from Start Menu and it will look like this



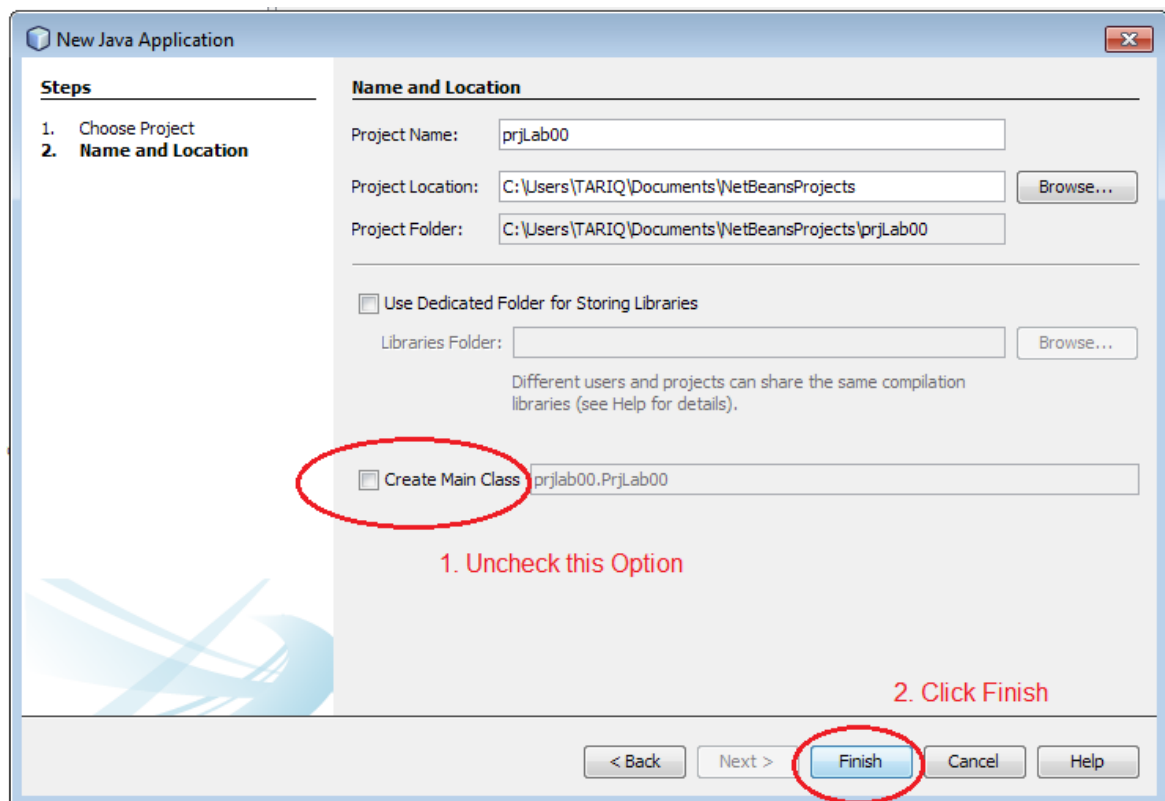
- In the IDE, choose File > New Project (Ctrl-Shift-N), as shown in the figure below.



- In the New Project wizard, expand the Java category and select Java Application as shown in the figure below. Then click Next.

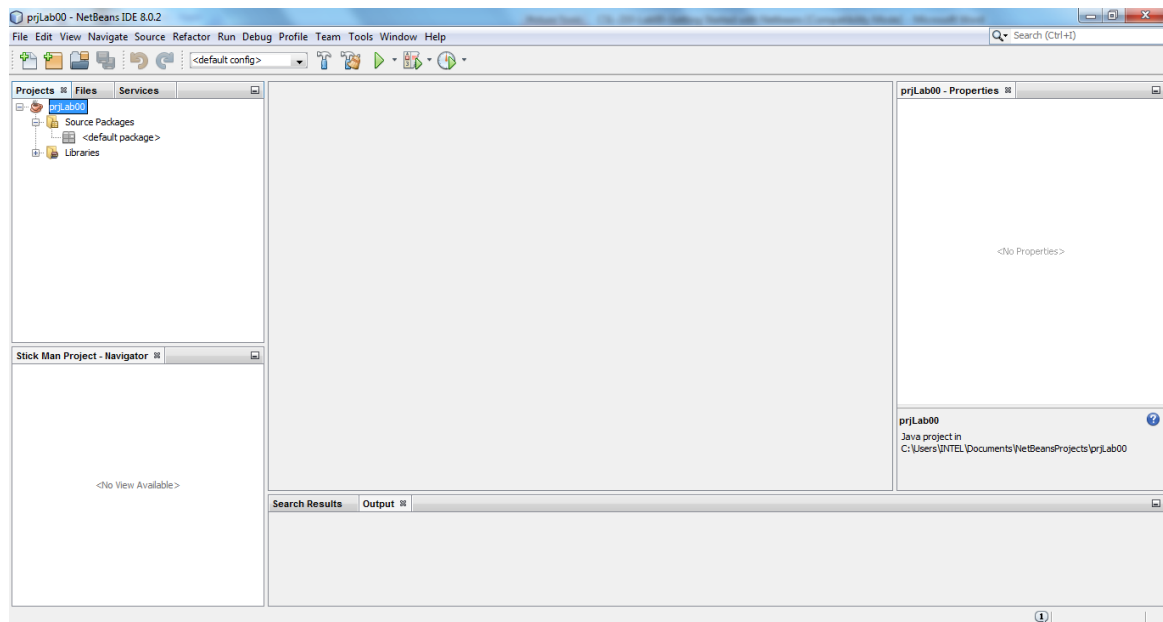


- In the Name and Location page of the wizard, do the following (as shown in the figure below):
- In the Project Name field, type prjLab00.
- Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
- Unselect the "Create Main Class".



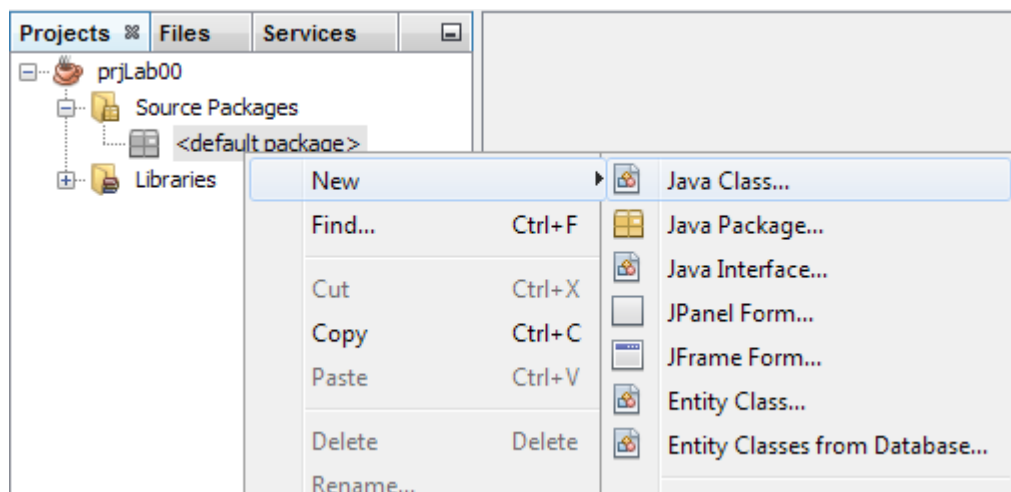
- Click Finish.
- The project is created and opened in the IDE. You should see the following components:

- The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.

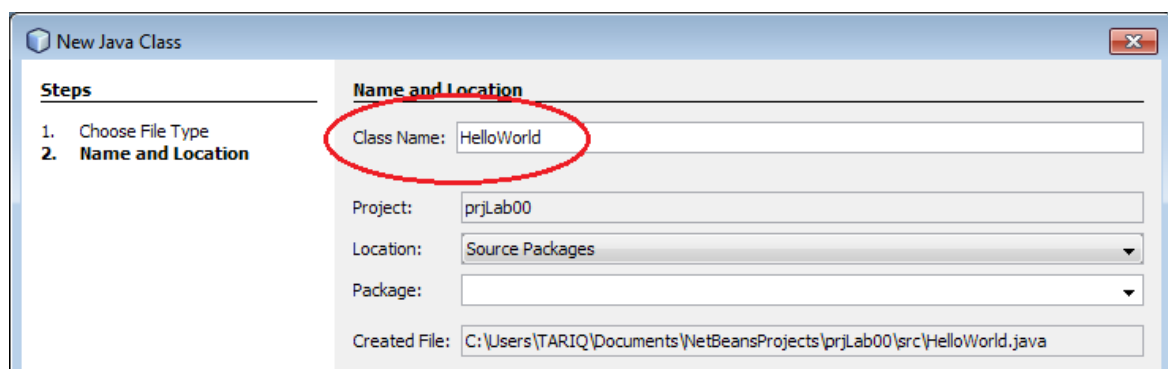


How to add your main class

Right click on default package and then select New and click on Java Class.



Click on Java Class you will see a New Java Class Wizard. Key in the Class Name HelloWorld and click Finish.



Adding Code to the Generated Source File:

Because you have left the Create Main Class checkbox unselected in the New Project wizard, the IDE has not created. You can add the "Hello World!" message to the skeleton code by replacing the line:

```
// TODO code application logic here
```

with the line:

```
System.out.println("Hello World!");
```

Save the change by choosing File > Save.

The file should look something like the following:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author TARIQ
 */
public class HelloWorld {

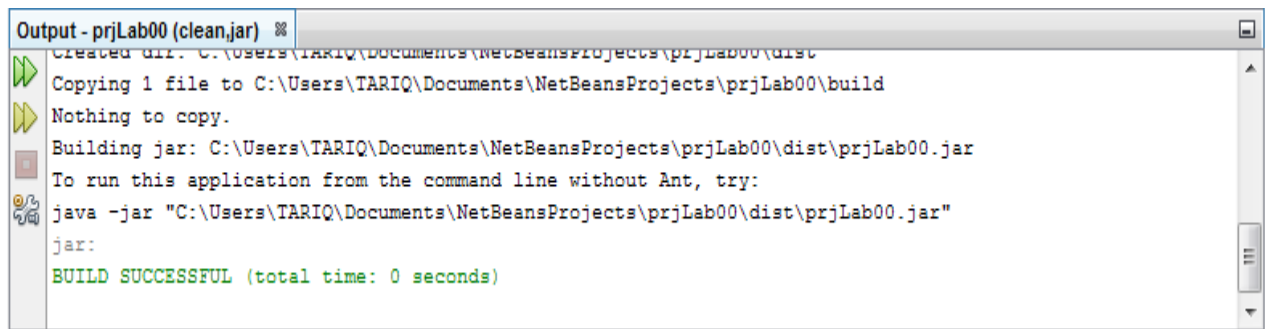
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Compiling the Source File:

To compile your source file, choose Build > Build Main Project (F11) from the IDE's main menu.

You can view the output of the build process by choosing Window > Output > Output.

The Output window opens and displays output similar to what you see in the following figure.



```
Output - prjLab00 (clean.jar)
Created dir: C:\Users\TARIQ\Documents\NetBeansProjects\prjLab00\dist
Copying 1 file to C:\Users\TARIQ\Documents\NetBeansProjects\prjLab00\build
Nothing to copy.
Building jar: C:\Users\TARIQ\Documents\NetBeansProjects\prjLab00\dist\prjLab00.jar
To run this application from the command line without Ant, try:
java -jar "C:\Users\TARIQ\Documents\NetBeansProjects\prjLab00\dist\prjLab00.jar"
jar:
BUILD SUCCESSFUL (total time: 0 seconds)
```

- If the build output concludes with the statement BUILD SUCCESSFUL, congratulations! You have successfully compiled your program!
- If the build output concludes with the statement BUILD FAILED, you probably have a syntax error in your code. Errors are reported in the Output window as hyper-linked text. Click such a hyper-link to navigate to the source of an error. You can then fix the error and once again choose Build > Build Main Project.

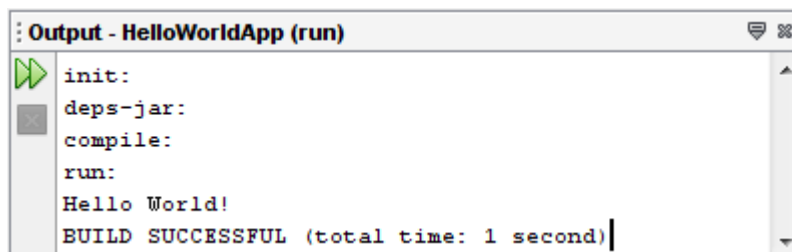
Now that you have built the project, you can run your program.

Running the Program:



Click to run the project

- From the IDE's menu bar, choose Run > Run Main Project (F6) OR
- The next figure shows what you should now see.



```
Output - HelloWorldApp (run)
init:
deps-jar:
compile:
run:
Hello World!
BUILD SUCCESSFUL (total time: 1 second)
```

Congratulations! Your program works!

2. Comments

Comments are English sentences inserted in a program to explain its purpose.

We use comments to explain the purpose of a class, a method or a variable.

There are two types of comments:

1) Block comment

```
/* ... */
```

- Comments are ignored by the compiler.
- Any message enclosed inside `/* ... */` is treated as a comment.
- This type of comment can span more than one line.

2) Line comment

```
// ...
```

- Any message following double forward slash, `//`, up to the end of the line is also considered a comment.
- **Notice that** this type of comment *cannot* span more than one line.
- It is often used to comment variables.

You are expected to:

- Write brief information about yourself at the beginning of each program,
- Write comments explaining each program you write in this course, and
- Comment every variable whose purpose is not obvious from its name.

3. Indentation

Indentation involves using *tabs*, *spaces*, and *blank lines* to visually group related statements together.

You shall:

- Push the statements inside a class, method, or structural statements such as `if`, `while`,... etc. by at least three or four spaces or a tab character;
- Separate between variable declarations and method declarations by a blank line; and
- Separate adjacent methods by a blank line.

4. Name Style

In programs, variable names should *clearly suggest their meanings*. This is called the **naming style**. Good naming style makes your program easier to read and easier to correct errors.

For example, the *A* coefficient of an equation should be called: `aCoefficient`, `coefficient_A`, or similar; but never `a`, `b`, `x`, or `z`.

The use of abbreviations is also discouraged.

5. Recognizing Syntax Errors

When you make *syntax* errors in your program the *compiler* gives error messages and does not create the *bytecode* file. It saves time and frustration to learn what some of these messages are and what they mean. Unfortunately, at this stage in the game many of the messages will not be meaningful except to let you know where the first error occurred. Your only choice is to carefully study your program to find the error.

Correcting Syntax Errors

Some things to remember:

- ✓ Java is *case sensitive*, so, for example, the identifiers `public`, `Public`, and `PUBLIC` are all considered different. For reserved words such as `public` and `void` and previously defined identifiers such as `String` and `System`, you have to get the case right.

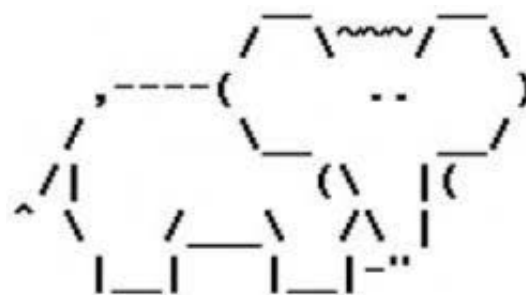
- ✓ When the compiler lists lots of errors, fix the first one (or few) and then recompile—often the later errors aren't really errors in the program, they just indicate that the compiler is confused from earlier errors.
- ✓ Always remember to close opened brackets, braces, and quotes.
- ✓ It is always important to remember to end every statement in Java with a *semicolon* ';'.
- ✓ Read the error messages carefully, and note what line numbers they refer to. Often the messages are helpful, but even when they aren't, the line numbers usually are.
- ✓ When the program compiles cleanly, run it.

Exercises

Exercise 1

(*Elephant.java*)

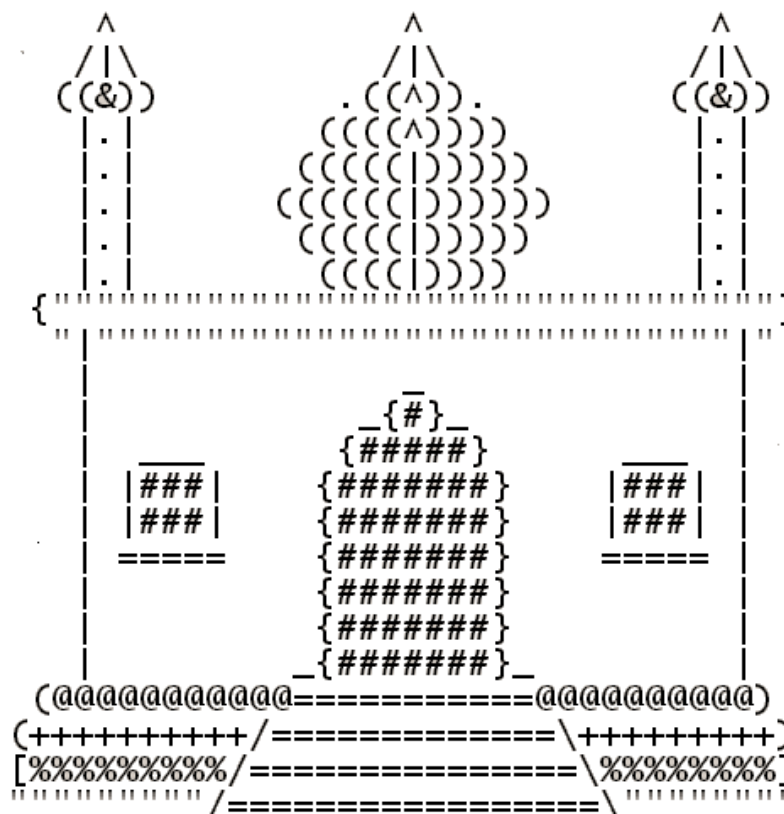
Write a program that prints an elephant similar to the following:



Exercise 2

(*Mosque.java*)

Write a program that prints a mosque, similar to the following:



Exercise 3

(Arithmetic.java)

Type-in the following example, which receives the input of two integer numbers and compute the sum, difference and product. Compile and run this program.

```
/*
 * Compute the sum, difference, and product of two integer numbers.
 */
import java.util.Scanner;
public class Arithmetic {
    /* Main method */
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);
        int number1; // 1st integer variable
        int number2; // 2nd integer variable
        System.out.print("Enter First Number: ");
        number1 = input.nextInt();
        System.out.print("Enter Second Number: ");
        number2 = input.nextInt();
        System.out.println("Sum is: " + (number1 + number2));
        System.out.println("Difference is: " + (number1 - number2));
        System.out.println("Product is: " + (number1 * number2));
    } //Main method ends
} // class ends
```

Exercise 4

(TirePressure.java)

Automobile Tire Pressure: $P = 0.37m(T + 460)/V$

P = pressure in psi.

V = volume in cubic feet

m = mass of air in pounds

T = temperature in Fahrenheit

Exercise 5

(Cookies.java)

There are 12 cookies per box (sold at \$1.14) and 24 boxes per carton. Left over boxes are sold for 57¢. Remaining cookies are given away free. Given the number of cookies produced, determine the number of boxes, cartons, left over boxes and the total money made.

Exercise 6

(PullyFormulas.java)

Pulley formulas

- a) calculate the speed of one pulley if there are 2 pulleys connected with a belt:

$$\text{RPM2} = \text{diameter1} / \text{diameter2} * \text{RPM1}$$

- b) calculate the amount of weight that can be lifted with a multiple pulley system:

$$\text{weight lifted} = \text{force exerted} * \text{number of up ropes}$$

Exercise 7

(BMI.java)

The body mass index (BMI), or Quetelet index, is a heuristic proxy for human body fat based on an individual's weight and height. BMI does not actually measure the percentage of body fat. We will be building a BMI calculator method. Body mass index (BMI) is computed using the formula,

$$\text{BMI} = \frac{\text{mass}(\text{lb})}{(\text{height}(\text{in}))^2} \times 703$$

Where mass is the subject's weight in pounds (lb) and height is the height in inches (in). The value 703 is a factor to convert BMI to a value that matches the original BMI calculations done in metric units (i.e. kilograms-meters).