**Simple programs and their meanings: Program 2**

import java.io.\*;

class Circle1

{

public static void main(String args[])throws IOException

{

// put your code here

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

float Radius;

double Area,Circumference;

System.out.println("\u000c");//this must be written to clear the previous output

System.out.print("Enter the radius:");

Radius=Float.parseFloat(in.readLine());

System.out.println();

Area=Math.PI\*Radius\*Radius;

Circumference=2\*Math.PI\*Radius;

System.out.println("The area of the circle is:"+Area);

System.out.println("The circumference of the circle is:"+Circumference);

in.close();

}

}

Line by line meaning of the code:

**First line:** import java.io.\*;

**Java.io** is a package which we want to include in our program. To include any package or any features (i.e. any class… since package puts related class together) we must use the keyword **import**.

To include all the features of the package java.io. we must write like: **import java.io.\*;** i.e. putting a star after “java.io.”. Otherwise we could mention the features of the **java.io.** package separately.

Like:

import java.io.IOException;

import java.io.BufferedReader;

import java.io.InputStreamReader;

Since, only these three features of the java.io. package are used in this particular program.

**Second line:** class Circle1

This line declares a class and Circle1 is an identifier that specifies the name of the class.

**Third line:{**

An opening brace. This denotes the scope for the class. i.e. the data members of the class the functions/methods which operate on the data members must be declared within the opening and closing braces of that class which defines the scope of it. (Scope of the class and scope of the variables and the functions which are local to the class)

**Fourth line:** public static void main(String args[])throws IOException

We have already discussed the use of public, static & void keywords for the main method. Now, I will tell the significance of string args []. String is an in build datatype for java and args [] is an array of String datatype. When the main function has only small no. of variables to deal with, which must be specified by the user then we can take those inputs with the help of args [] array. This is called taking command-line arguments.

|  |  |
| --- | --- |
|  | IOException is a [checked exception](http://en.wikipedia.org/wiki/Exception_handling#Checked_exceptions). You must either catch it, or throw it to your calling method. |

It is caused by general I/O failures. (As a beginner you don’t need to know the details of it. What you need to know is that if you are taking user input with the help of BufferedReader or DataInputStream you must throw IOException. Otherwise the code will not be compiled (i.e. it will generate a compilation error) ).

**Fifth line:{**

Another opening braces… which is for defining the scope of main. It must be ended with a closing brace.

Now, all the variables which will be defined within the main method, and all the lines written within it, are local to the main. i.e. scope is main

**Sixth line://put you code here**

This is called comment line. Comment lines are extracted from the program before compiling. So, it does not have an impact in the generated bytecode (generated by compiler) as well as generated machine code(by the java interpreter)

**Seventh line:** BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

Now **BufferedReader** is a class inside **java.io.** package and to reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines. Now, here **“in”** is the object of **BufferedReader** which we create to use the methods of **BufferedReader** . Now, to initialize an object of a class, we must use its constructor. **BufferedReader** class has two types of constructor:

|  |  |
| --- | --- |
| S.N | Constructor and Description |
| 1 | **BufferedReader(Reader in)**  This creates a buffering character-input stream that uses a default-sized input buffer. |
| 2 | **BufferedReader(Reader in, int sz)**  This creates a buffering character-input stream that uses an input buffer of the specified size. (sz) |

Here, we initialize the object of **BufferedReader** using the first type of constructor. But the problem with that –it cannot wrap an **InputStream** directly. It wraps another Reader. That’s why, we have to wrap the **InputStream** indirectly via **InputStreamReader**. The concept is like this:

**InputStreamReader** allows you to associate a stream that reads from the specified input (in this case the standard input), so now we have a stream.

**BufferedReader** is an "abstraction" to help you to work with streams. For example, it implements **readLine** instead of reading character by character until you find a '\n' to get the whole line. It just returns a String after this process.

**Seventh line:** float Radius;

It declares a variable of datatype float and the variable’s name is Radius (Radius is actually an identifier which associates with the newly declared variable)

**Eighth line:** double Area,Circumference;

This line declares two variables of datatype double and the name of those two variables are: Area and Circumference.

**Note that:** It is a good programming practice to give meaningful variable names.

**Ninth line:** System.out.println("\u000c");

**System** is a method under/in java.lang package. Now, java.lang package is by default included to all the java programs. So, it needs not to be explicitly specified.

Java.lang

Class System

Java.lang.object

Java.lang.System

public final class System extends object

Now, the System class (of java.lang package) contains several useful class fields and methods. Among the facilities provided by the System class are standard input, standard output and error output streams. Here, we use the standard output facility.

**System.out.println(“\u000c”);**

By doing this, we are first clearing the previous outputs and ensures that the next line which will be printed in the outputbuffer will be printed from a new, fresh line.

**Tenth line:** System.out.print("Enter the radius:");

This line is to print the string **“Enter the radius”** in the outputbuffer to ask the user to enter the radius for calculating the **area** and **circumference** of the corresponding **circle**.

**Eleventh line:** Radius=Float.parseFloat(in.readLine());

Here, we use the readLine() method of BufferedReader to read a line of text (user input). Now, since readLine() is not a static method it must be used via an object of BufferedReader. That’s why we need to create the object in of BufferedReader. Now, the method readLine() ‘s return type is String. i.e. it converts the fetched line of text from inputbuffer to a String. But , radius is variable of float datatype. That means we need to convert it to the float datatype. This is done by the parseFloat method of Float class (Float is a class within java.lang package which helps us to deal with the operations on float datatype.) Also note that, parseFloat is a static method. So, we can use the parseFloat method without creating an object of Float class. We can directly use the parseFloat method like this: Float.parseFloat(arguments).

**Note that:** Float is a wrapper class. It wraps predefined basic type float.

* Float.parseFloat method

static float parseFloat(String s)

This method returns a new float initialized to the value represented by the specified String , as performed by the valueOf method of class Float.

**Twelfth line:** System.out.println();

This is to end the current line in the outputbuffer and ensure that the next line which will be printed in outputbuffer will be printed in a new line.

**Thirteenth line:** Area=Math.PI\*Radius\*Radius;

To get the value of the π (Since area of a circle is **π×(radius)2**) we have to use the static method Pi of Math class of java.lang package. Now, since return type of the static method Pi is double the variable whose value is calculating using the Pi also must be variable of double datatype. Otherwise, there will be a compilation error as “Possible loss of precision”.

**Fourteenth line:** Circumference=2\*Math.PI\*Radius;

This line is to calculate the value of **Circumference** of a circle. Note that, **Circumference** must also be a variable of datatype double.

**Fifteenth line:** System.out.println("The area of the circle is:"+Area);

First the String (“The area of the circle is:”) will be printed in a new line then the value of the area after the “:”.

Now actually what happens: System.out.print(args) and System.out.println(args) both can only print a string (a character sequence) in the outputbuffer. So, first the value of the Area will be implicitly converted to string then using the **“+”** operator it is concatenated (added) to the string “The area of the circle is:”). Then, it is printed in the outputbuffer through the method System.output.println().

**Sixteenth line:** System.out.println("The circumference of the circle is:"+Circumference);

This line is to print the string **“The circumference of the circle is:”** following by the value of the **Circumference**.

**Seventeenth line:** in.close();

The method close() of **BufferedReader** class**,** closes the stream and releases any system resources associated with it. It is a good programming habit to close the stream after all its use.

**Eighteenth line:}**

A closing brace to end the scope of the main method

**Nineteenth line:}**

Another closing brace to end the scope of the class named **“Circle1”**