EXCERCISES.

## INTRODUCTION TO JAVA

### Hello World

The purpose of this exercise is to verify that you know how to run a basic Java application.

#### Exercise steps:

1. Create a Java package called exercises .
2. Inside the exercises package, create another package (subpackage) called java
3. Create a Java class called Exercise1 inside the java package.
4. Insert a main()method inside the Excercise1 class.
5. Inside the main() method, insert this statement:   
   System.out.println("Hello World");
6. Compile and run the main() method of the Excercises1 class.

## VARIABLES AND OPERATORS

### Using Variables

The purpose of this exercise is to know how to declare and use variables in Java.

#### Exercise:

1. How to declare primitive and non-primitive data types

### Variables Scopes

*Exercise:*

1. Declare a variable with a Class Level scope.

2. Declare a method scope variable

3. Declare a variable with a Loop level Scope

### Primitive Types

1. Declare correctly all primitive Java variables:

### Operators

1. Mention and declare the five types of operators allowed in Java:

### Strings

1. Declare two different ways to create Strings

### Arrays

* 1. Declare the different ways to initialize and assign a value to a one dimensional Arrays.

## CLASSES

### Creating a Class.

1. Create in code a simple and small ecosystem based on animal object and their different types of animal classes and define a common behavior between them.

### Using methods

1. Declare an Overriding Method with Java code.

2. Declare an overloading method with Java code.

### Constructors

1. Making a constructor.

2. Overloading a constructor.

3. Using the default constructor.

### Access Modifiers

1. Write code to show how to apply access modifiers in Java.

## CONTROL STATEMENTS

### Transfer of control Statements

1. From following code, provide the correct output;

int th = 0;

int daf = 4;

for (int i=1; i<25;i++){

th = ++th;

if(th > daf)

**break**;

System.out.println(“Fishing for hour ”+ i);

}

2. From following code, provide the correct output:

Int tdc = 0;

Int daf = 5;

for (int i=1; i<8; ++i){

System.out.println(“\nDay ”+ i + “: camping”);

tdc = ++tdc;

if(tdc > daf)

**continue**;

System.out.println(“and Fishing“);

}

2. Decode a code with a *return* statement.

### Flow control Statements.

1. What is the value of b at the end of this code:

foo () {

int b = 1;

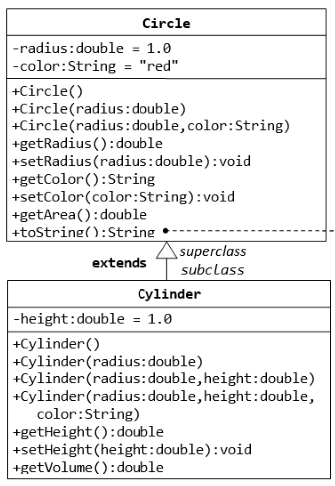
b++;

}

## INHERITANCE AND COMPOSITION

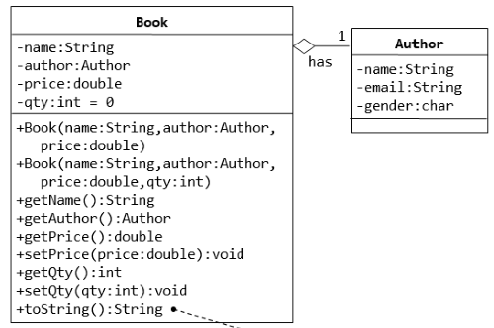
### Inheritance

1. Create the code based on the following Class Diagram (just consider the constructor method creation):



### Composition

1. Write the Book class (which uses the Author class written earlier). Also write a test driver called TestBook to test all the public methods in the class Book. Take Note that you have to construct an instance of Author before you can construct an instance of Book



### Polymorphism

1. Write a test class to test these statements involving polymorphism and explain the outputs. Some statements may trigger compilation errors. Explain the errors, if any.



### Interfaces.

1. Write a class that implements the CharSequence interface found in the java.lang package. Your implementation should return the string backwards. Select one of the sentences from this book to use as the data. Write a small main method to test your class; make sure to call all four methods.

## GENERICS

1. Elimiation of casts. Re-write the following code to use generics, the code does not require casting:

List list = new ArrayList();

list.add("hello");

String s = (String) list.get(0);