A

Expand JDK acronym.

JDK – Java Development Kit

A

What is the purpose of JDK?

JDK contains the minimum software needed for Java development.

A

Name four key commands of JDK.

R

Key commands are:

`javac` `java` `jar` `javadoc`

A

What does the `javac` command do?

`javac` command converts .java source files into .class bytecode that can be understood by Java Virtual Machine.

A

What does the `java` command do?

`java` command executes the program by launching the Java Virtual Machine.

A

What does the `jar` command do?

`jar` command packages .class files together.

A

What does the `javadoc` command do?

`javadoc` command generates documentation.

A

Expand JVM acronym.

JVM – Java Virtual Machine

A

Which files can be executed by the Java Virtual Machine? .java files or .class files?

Java Virtual Machine understand bytecode, so it works with .class files.

A

Expand IDE acronym.

IDE – Integrated Development Environment

A

Name three most common Java IDEs.

Common Java IDEs include:

Eclipse, IntelliJ IDEA, Visual Studio Code

A

Expand JRE acronym.

JRE – Java Runtime Environment

A

What is the difference between JDK and JRE?

R

JRE is a subset of JDK that can be used for running a program but cannot compile one.

A

What is the difference between .java and .class files?

.java files store source code (as written by the programmer). .class files store bytecode which can be understood by the Java Virtual Machine.

A

Is JVM part of JDK or JRE?

R

The JVM is included in both the JDK and the JRE.

A

Is compiler part of JDK or JRE?

The compiler is included in JDK only.

A

For Java 17 on the Oracle website, is JDK or JRE available for download?

For version 17, only a JDK is available.

A

How often does Oracle release a new version of Java?

New versions of Java are released every six months (in March and September).

A

Expand LTS acronym.

LTS – Long-Term Support

A

What is Java LTS version?

R

LTS versions of Java are versions supported by Oracle for many years. It is recommended that you create programs in these versions.

A

Name all Java LTS versions (up to Java 24).

Java LTS versions:

Java 8, Java 11, Java 17, Java 21

A

What command can be used to check current version of Java on the command line?

The correct command is:

```
java --version
```

What is an object?

R

An object is a runtime instance of a class in memory. An object is often referred to as an instance since it represents a single representation of the class.

What is a reference?

R

A reference is a variable that points to an object.

A

Name two primary elements of Java classes.

Primary elements of Java classes are fields and methods. Together these are called the members of the class.

A

How can you create a public class named 'Animal' without any members?

Syntax is as follows:

```
public class Animal {

}
```

A

How can you create a class named 'Person' with one private field of type String named 'surname'?

Syntax is as followed:

```
class Person {
    private String surname;
}
```

A

How can you create a class named 'Color' with one public method named 'isTransparent' that always returns `true`?

Syntax is as followed:

```
class Color {
   public boolean isTransparent() {
      return true;
   }
}
```

What makes a method signature?

Method signature includes method name and parameter types.

Write method signature of a following method:

```
public int increment(int x) {
    return x + 1;
}
```

Method signature is as follows:

```
increment(int)
```

A

Write method signature of a following method:

```
private static void print() {
    System.out.println("Test");
}
```

Method signature is as follows:

```
print()
```

A

Write method signature of a following method:

```
private long add(int x, long y) {
    return x + y;
}
```

Method signature is as follows:

```
add(int, long)
```

A

Name the following comment type:

```
//my comment
```

This comment type is named *single-line comment*.

A

Name the following comment type:

```
/*
 my comment
*/
```

This comment type is named *multiple-line comment.*

A

Name the following comment type:

```
/**
 my comment
*/
```

This comment type is named *Javadoc comment*.

A

Will this comment compile? If so, what type of comment is it?

```
/*
* my comment
*/
```

Yes, the syntax is correct. This is a multiple-line comment.

A

Will this comment compile? If so, what type of comment is it?

```
/*
// who am I?
*/
```

Yes, the syntax is correct. This is a multiple-line comment.

A

Will this comment compile? If so, what type of comment is it?

```
// // make it double
```

Yes, the syntax is correct. This is a single-line comment.

A

Will this comment compile? If so, what type of comment is it?

```
// /* undecided */
```

R

Yes, the syntax is correct. This is a single-line comment.

A

Will this comment compile? If so, what type of comment is it?

```
/* one line */
```

R

Yes, the syntax is correct. This is a multiple-line comment (even though it only takes up one line).

A

Will this comment compile? If so, what type of comment is it?

```
/*
/* many stars */
*/
```

R

No, the syntax is invalid. The multiple-line comment ends with first '*/', so second occurence will not compile.

A

Will this comment compile? If so, what type of comment is it?

```
/**
* documentation
**/
```

Yes, the syntax is correct. This is a javadoc comment.

A

Will this comment compile? If so, what type of comment is it?

```
/***/
 documentation
*/
```

R

No, the syntax is wrong. Compiler see a single multiple-line comment (from /* to */). Directly after there is a single * symbol, which cannot be compiled.

A

Will this comment compile? If so, what type of comment is it?

```
/**/// comment
```

R

Yes, the syntax is correct. There are two comments in this example. First we have multiple-line comment (without contents), and then directly after we have single-line comment.

What is a top-level class?

R

A top-level class is a class that can be defined independently within a source file (so it is not a part of another class).

A

Must top-level class be public?

R

No, top-level class is not required to be public.

A

Can a single source file contain multiple top-level classes?

R

Yes, a single source file can contain multiple

top-level classes, but only one of them can by public.

A

A Java source file contains a top-level public class named 'MyClass'. How can you name this source file?

The source file must be named 'MyClass.java'.

A

A Java source file is named 'MyFile.java' and contains a top-level public class. How can you name this class?

The class must be named 'MyFile'.

A

Where does Java program execution start?

R

Program execution begins with a call to the main() method of the class that is being run.

A

Write program entry-point method with an empty body.

Correct syntax is:

```
public static void main(String[] args) {
}
```

You can also write `String...` as a type of parameter.

A

Is this method a valid entry-point into the program?

```
public static void main(String args[]) {
}
```

Yes, the method can be used as an entry-point into the program.

A

Is this method a valid entry-point into the program?

```
public final static void main(String[] args) {
}
```

R

Yes, the method can be used as an entry-point into the program. The `main()` method can be final.

A

Is this method a valid entry-point into the program?

```
public static void main01(String args[]) {
}
```

R

No, the entry-point method must be named 'main'.

A

Is this method a valid entry-point into the program?

```
public static void main() {
}
```

R

No, the entry-point method must have a String[] parameter.

A

Is this method a valid entry-point into the program?

```
public static void main(final String[] args) {
}
```

R

Yes, the method can be used as an entry-point into the program. Parameter of the method can be marked as final.

A

Is this method a valid entry-point into the program?

```
private static void main(String[] args) {
}
```

R

No, the entry-point method into the program must be public.

A

Is this method a valid entry-point into the program?

```
public void main(String[] args) {
}
```

R

No, the entry-point method into the program must be static – Java will not create an instance of the class to run the program.

A

Is this method a valid entry-point into the program?

```
public static int main(String[] args) {
   return 0;
}
```

R

No, the entry-point method into the program must be of the void return type.

A

We have a public class named 'Zoo' inside a 'Zoo.java' file. Inside the class there is a valid main() method. How to launch the program, using this method as an entry-point?

We need to translate the Zoo.java file to the bytecode:

```
javac Zoo.java
```

Then we can launch the Zoo.class file:

```
java Zoo
```

A

We have a class named 'ZooManager' class inside a 'Zoo.java' file. Inside the class there is a valid main() method. How to launch the program, using this method as an entry-point?

R

It is not possible. Launching program through the 'Zoo' file will look for the public class named 'Zoo'. It will not find `main()` method inside another class in this file.

A

We have a valid source code inside a Bird.java file. Which command is a proper way to translate it into .class file.

a) `java Bird.java`

b) `java Bird`

c) `javac Bird.java`

d) `javac Bird`

R

Correct command is c) `javac Bird.java`

A

We have a valid bytecode inside a 'Bird.class' file. Which command is a proper way to launch the program through this file?

a) `java Bird.class`

b) `java Bird`

c) `javac Bird.class`

d) `javac Bird`

Correct command is b) `java Bird`

A

We have a valid source code inside a 'Bird.java' file. How to run this code using single command, without creating a .class file?

a) `java Bird.java`

b) `java Bird`

c) `javac Bird.java`

d) `javac Bird`

Correct command is a) `java Bird.java`

A

We have a valid source code inside a 'Bird.java' file. How to run this code using a single command, without creating a .class file? We want to pass two parameters: 'hello' and 'world'.

R

Correct command is:

```
java Bird.java hello world
```

A

We have a valid source code inside a 'Bird.java' file. How to run this code using a single command, without creating a .class file? We want to pass one parameter: 'hello world'.

Correct command is:

```
java Bird.java "hello world"
```

What is a package?

R

A package is a logical grouping for classes.

A

Write an import statement for class Random from package java.util.

R

Correct syntax is:

```
import java.util.Random;
```

A

Write an import statement for all classes from java.text package.

Correct syntax is:

```
import java.text.*;
```

A

Will this compile?

```
import java.util.Date;
class MyClass {
  private Date myDate;
}
```

R

Yes, the syntax is correct.

A

Will this compile?

```
import java.util.*;
class MyClass {
  private Date myDate;
}
```

R

Yes, the syntax is correct.

A

Will this compile?

```
import java.*;
class MyClass {
  private Date myDate;
}
```

No, wildcard imports do not bring in child packages.

A

While using wildcard import, does compiler import all the classes from the imported package?

R

No, compiler will import only those classes which are used by the code in the source file.

A

Which package does not require import statement to be used?

This package is java.lang.

A

Will this compile?

```
import java.text.Date;
class MyClass {
  private Date myDate;
}
```

R

No, there is no 'Date' class in java.text import.

A

Will this compile?

```
import java.util.Date;
import java.util.Date;
class MyClass {
  private Date myDate;
}
```

R

Yes, the syntax is correct, but one of the import statements is redundant.

A

Will this compile?

```
import java.util.Date;
import java.sql.Date;
class MyClass {
  private Date myDate;
}
```

No, compiler cannot decide which 'Date' class should be used.

A

Will this compile?

```
import java.util.*;
import java.sql.Date;
class MyClass {
  private Date myDate;
}
```

R

Yes, the syntax is correct. Compiler will use java.sql.Date class for variable myDate.

A

Will this compile?

```
class MyClass {
  private java.sql.Date myDate;
}
```

R

Yes, the syntax is correct. Import statement is not required in this scenario.

A

Will this compile?

```
import java.util.Date;
import java.sql.Date;
class MyClass {
  private java.sql.Date myDate;
}
```

R

No. Compiler will know which package should be used for myDate variable, but it is not allowed to use two import statements importing classes with the same name.

A

Will this compile?

```
import java.util.*;
import java.sql.*;
class MyClass {
  private java.sql.Date myDate;
}
```

R

Yes, the syntax is correct.

A

Classes 'Files' and 'Paths' are part of the java.nio.file package. Will this compile?

```
class MyClass {
  private Files myFiles;
  private Paths myPaths;
}
```

No, classes must be imported.

A

Classes 'Files' and 'Paths' are part of the java.nio.file package. Will this compile?

```
import java.nio.*;
class MyClass {
  private Files myFiles;
  private Paths myPaths;
}
```

R

No, importing java.nio does not allow us to use classes from its child package.

A

Classes 'Files' and 'Paths' are part of the java.nio.file package. Will this compile?

```
import java.nio.*.*;
class MyClass {
  private Files myFiles;
  private Paths myPaths;
}
```

R

No, the syntax *.* is invalid.

A

Classes 'Files' and 'Paths' are part of the java.nio.file package. Will this compile?

```
import java.nio.file.*;
class MyClass {
  private Files myFiles;
  private Paths myPaths;
}
```

Yes, the syntax is correct.

A

Classes 'Files' and 'Paths' are part of the java.nio.file package. Will this compile?

```
import java.nio.file.Files;
import java.nio.file.*;
class MyClass {
  private Files myFiles;
  private Paths myPaths;
}
```

Yes, the syntax is correct.

A

Classes 'Files' and 'Paths' are part of the java.nio.file package. Will this compile?

```
import java.nio.file.Files;
import java.nio.file.Paths;
class MyClass {
  private Files myFiles;
  private Paths myPaths;
}
```

Yes, the syntax is correct.

A

Classes 'Files' and 'Paths' are part of the java.nio.file package. Will this compile?

```
import java.nio.file.Files.*;
import java.nio.file.Paths.*;
class MyClass {
  private Files myFiles;
  private Paths myPaths;
}
```

R

No, the syntax is invalid.

A

What is a default package?

R

A default package is a special unnamed package, that should be used only for throwaway code. Classes inside the default package do not have a package statement.

A

Write a package statement for package named 'pack'.

R

Statement should be as follows:

```
package pack;
```

A

Write a package statement for package named 'dog' inside the package named 'animal'.

R

Statement should be as follows:

```
package animal.dog;
```

A

We have a class named 'MyClass', as seen below. Write an import statement to use this class.

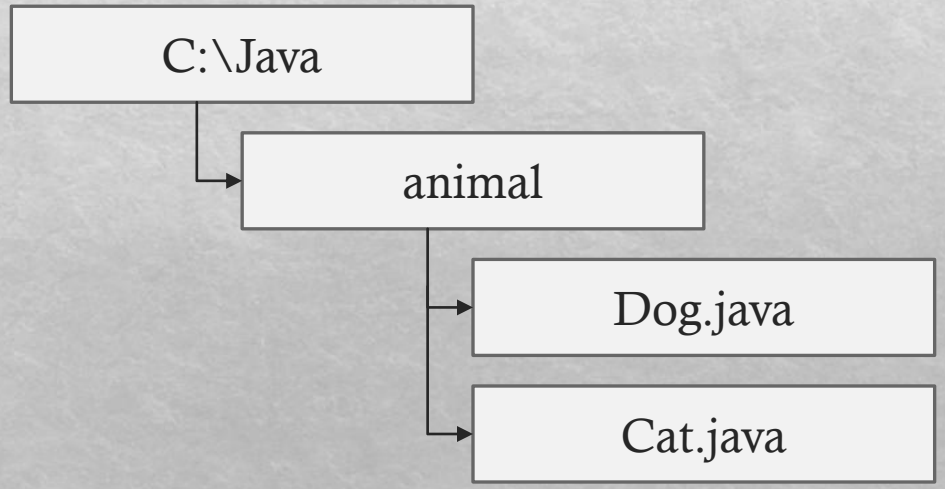
```
package a.b.c;

public class MyClass {}
```

Import statement should be as follows:

```
import a.b.c.MyClass;
```

A

We have a following files structure. Working directory is C:\Java. Write a command to compile the 'Dog.java' file.



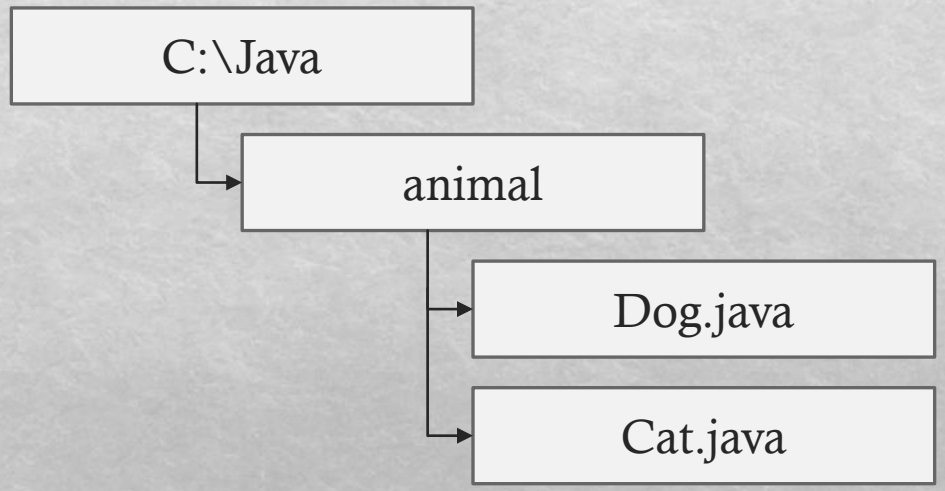
R

Command should be as follows:

```
javac animal/Dog.java;
```

A

We have a following files structure. Working directory is C:\Java. Will following command compile both .java files?



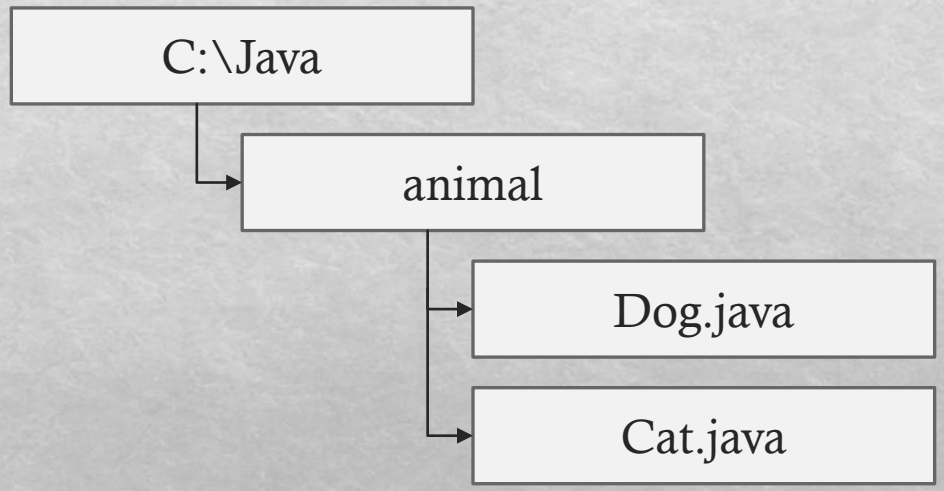

```
javac Dog.java Cat.java
```

R

No, the command is incorrect, because there are no .java files directly inside C:\Java folder.

A

We have a following files structure. Working directory is C:\Java. Will following command compile both .java files?



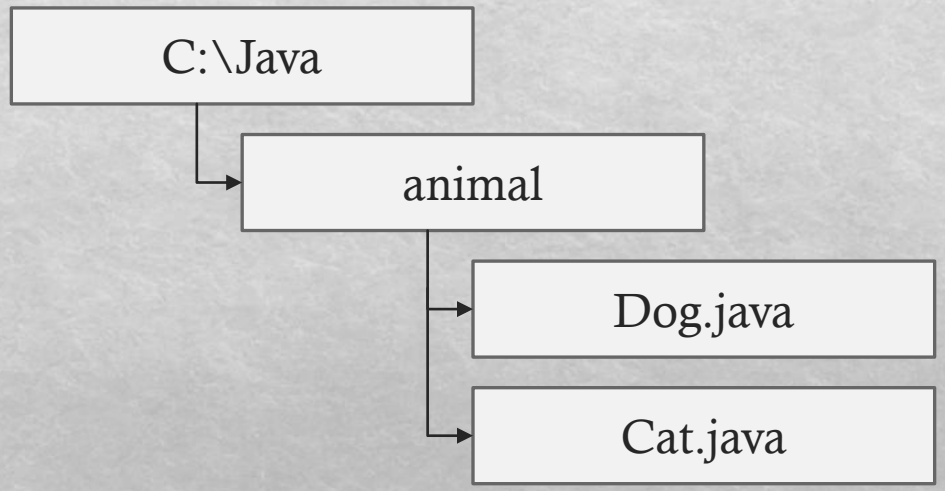

```
javac animal/Dog.java animal/Cat.java
```

R

Yes, the command is correct.

A

We have a following files structure. Working directory is C:\Java. Will following command compile both .java files?



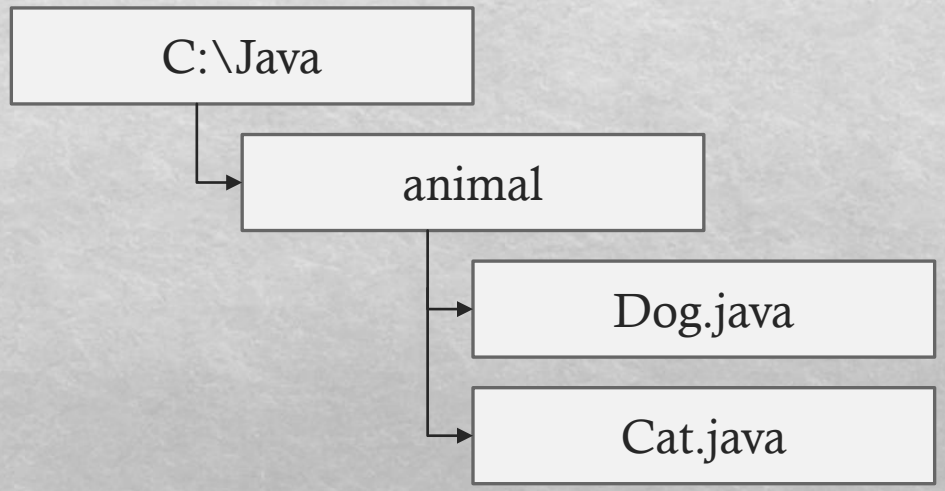

```
javac animal/Dog animal/Cat
```

R

No, while using javac command we must write files with .java extension.

A

We have a following files structure. Working directory is C:\Java. Will following command compile both .java files?



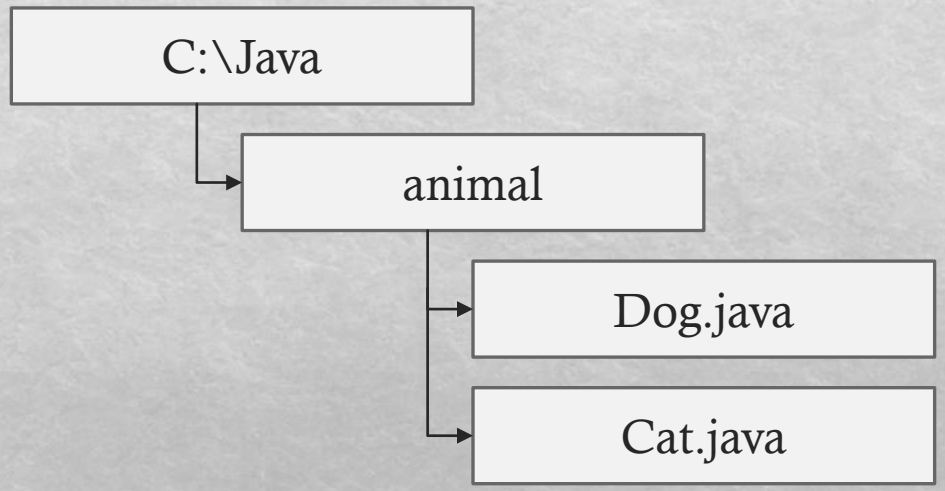

```
javac animal/*
```

R

Yes, the command is correct.

A

We have a following files structure. Working directory is C:\Java. Will following command compile both .java files?



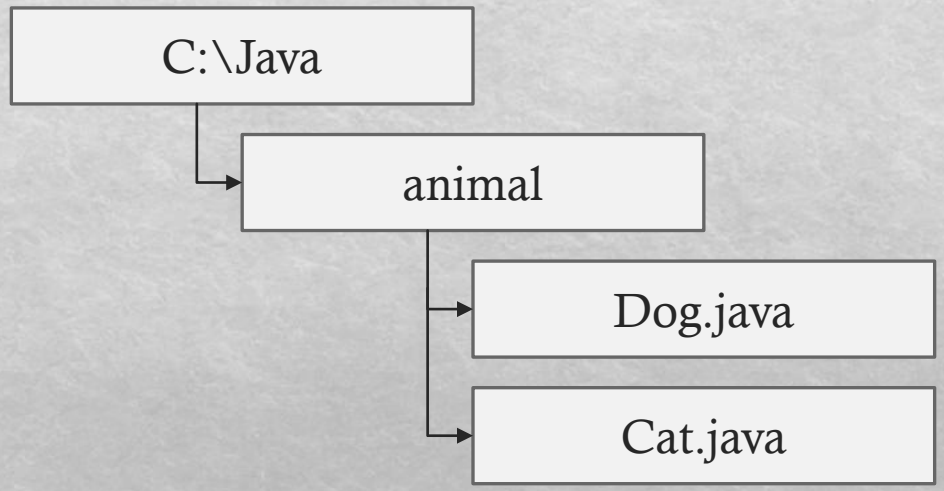

```
javac animal/*.java
```

R

Yes, the command is correct.

A

We have a following files structure. Working directory is C:\Java. Will following command compile both .java files?



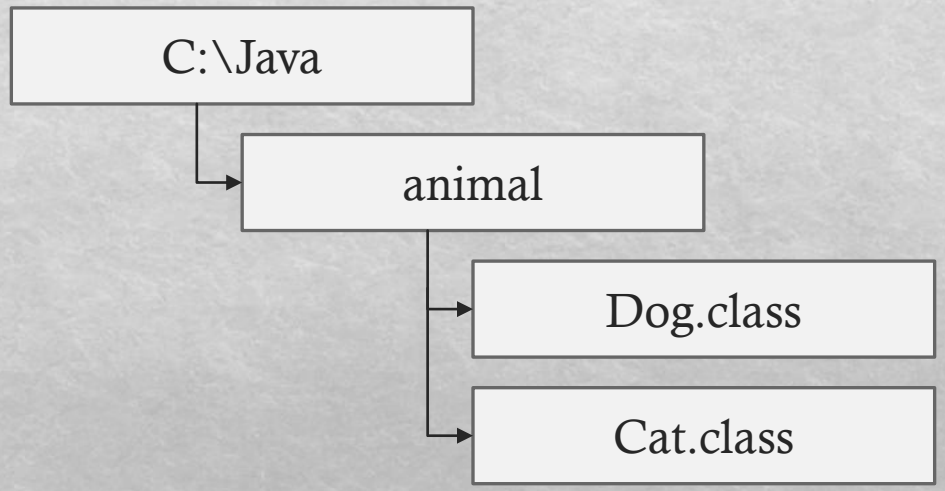

```
javac animal.Dog.java animal.Cat.java
```

R

No, while using javac command we must separate packages from classes with '/' symbol.

A

We have a following files structure. Working directory is C:\Java. Will following command launch only 'Dog.class' file?



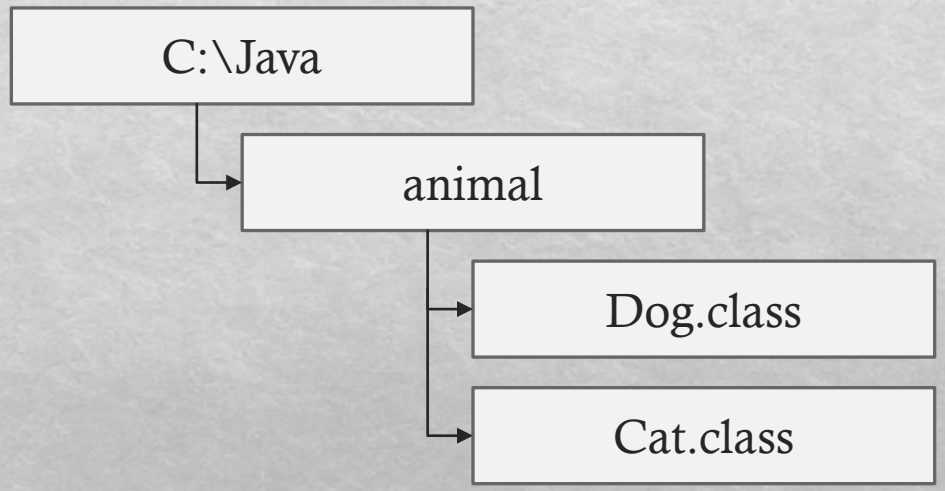

```
java Dog
```

R

No, the command is incorrect. We need to write the package name.

A

We have a following files structure. Working directory is C:\Java. Will following command launch only 'Dog.class' file?



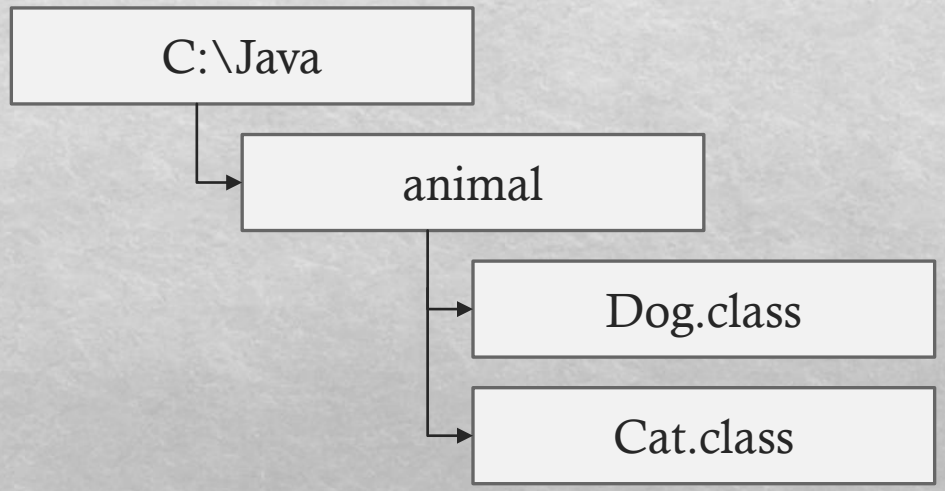

```
java animal.Dog
```

R

Yes, the command is correct.

A

We have a following files structure. Working directory is C:\Java. Will following command launch only 'Dog.class' file?



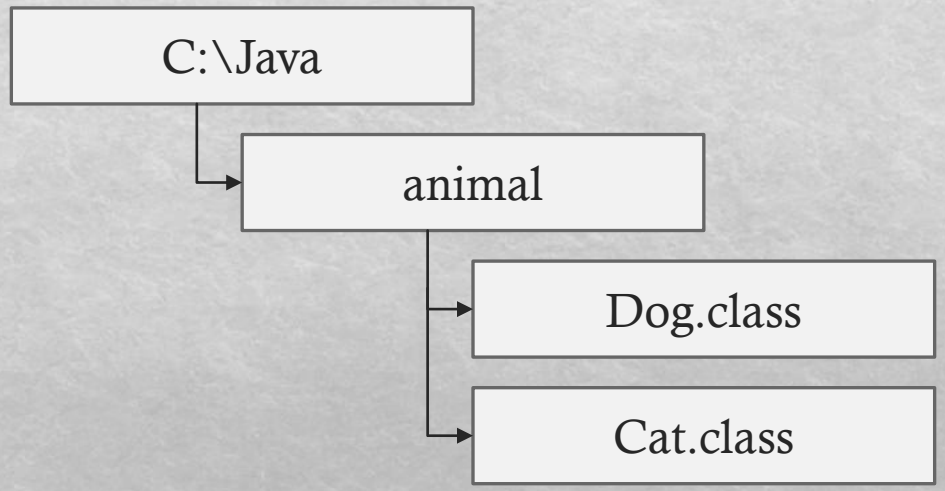

```
java animal.Dog.class
```

R

No, the command is incorrect. While using java command we cannot write file extension.

A

We have a following files structure. Working directory is C:\Java. Will following command launch only 'Dog.class' file?



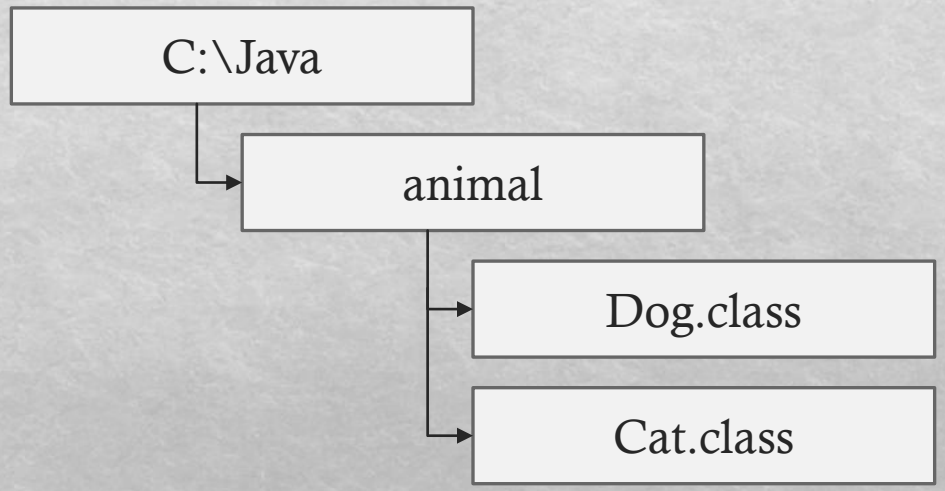

```
java animal/Dog
```

R

Yes, the command is correct, although while using java command it is better practice to use '. ' symbol to separate packages and classes.

A

We have a following files structure. Working directory is C:\Java. Will following command launch only 'Dog.class' file?




```
java animal.Dog animal.Cat
```

Yes, the command will launch 'Dog.class' only. 'animal.Cat' will be used as an argument for 'Dog' class main method.