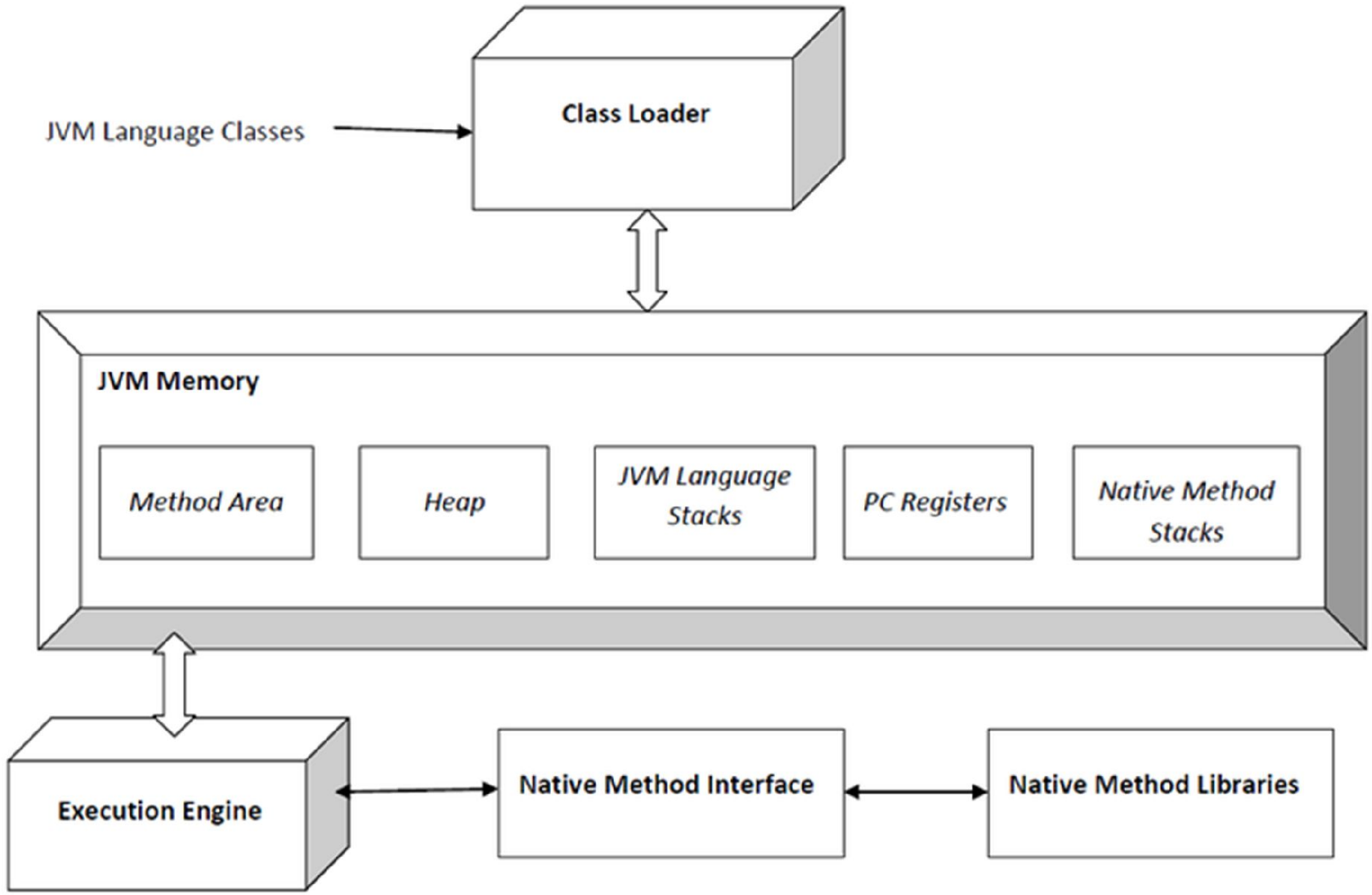
1. Java Basics(The JVM, Imports, Packages, and Class definitions)

How Java Works



The JVM, JRE, JDK

JVM (Java Virtual Machine):

- Think of the **JVM** as a *magic box* that runs your Java programs.
- When you write a Java program, it gets converted into something called **bytecode** (which the computer doesn't directly understand).
- The JVM's job is to take this bytecode and convert it into instructions your computer's hardware can understand
- So, the JVM is responsible for **executing** your Java programs.

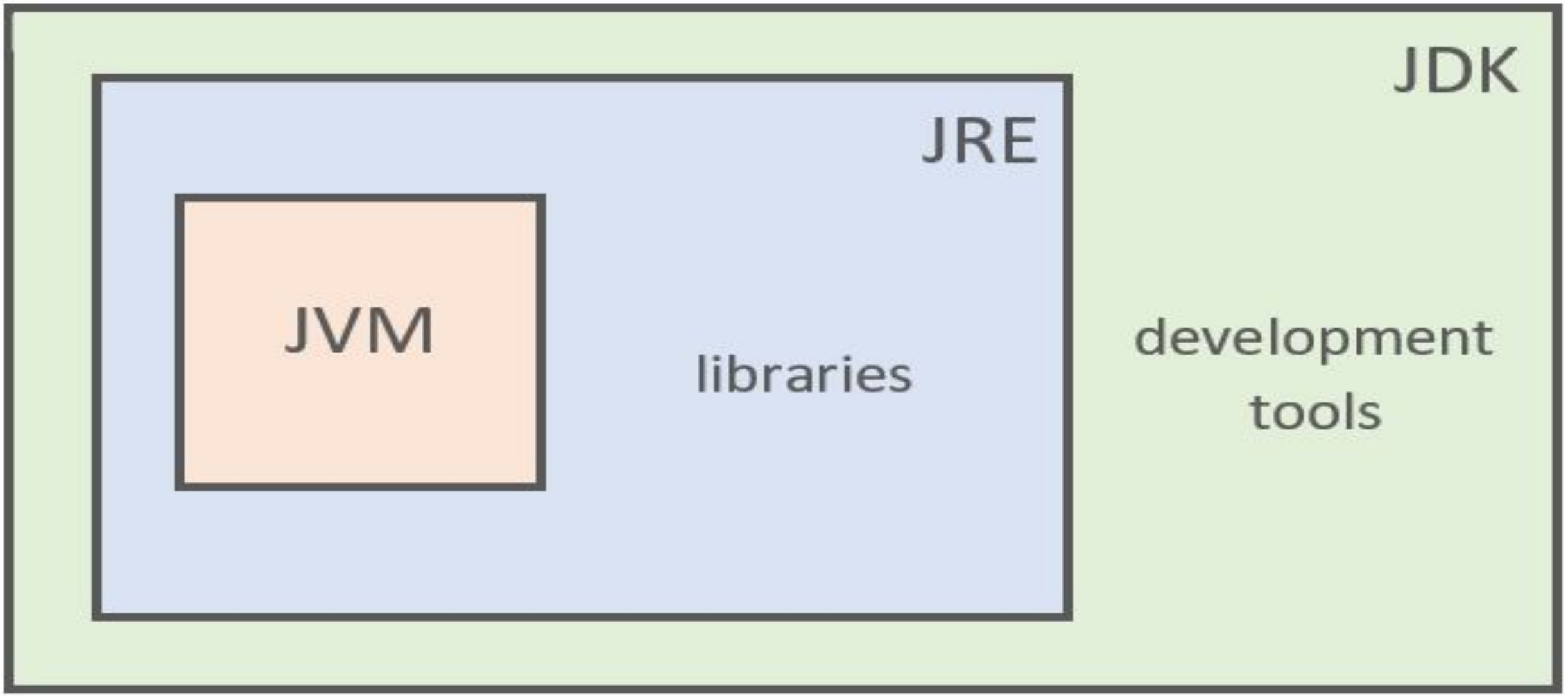
JRE (Java Runtime Environment):

- The **JRE** is like the *toolkit* that helps the JVM do its job.
- It includes the **JVM** and also the **libraries** (pre-built code) that Java programs need to run.
- If you just want to **run** Java applications (but not write them), you only need the JRE installed on your computer.

JDK (Java Development Kit):

- The **JDK** is the *complete package* for people who want to **write** and **develop** Java programs.
- It includes everything the JRE has (JVM + libraries), **plus** some extra tools (like a **compiler**) that turn your Java code into bytecode.
- If you want to create Java programs, you need the JDK.

The JVM, JRE, JDK



Structure of a Java File

package statement

import statements

// Any comments

Class Declaration{

  variables

  // maybe more comments

  constructors

  methods

  nested classes

  nested interfaces

  enum

  }

Not included in the Java SE 8 Programmer I Exam

Example File

```
// Package declaration
package com.example;

// Import statements
import java.util.ArrayList;
import java.util.List;

// Class declaration
public class MyClass {

    // Variables
    private String name;
    private int age;
    private List<String> hobbies;

    // Method to add a hobby
    public void addHobby(String hobby) {
        hobbies.add(hobby);
    }

    // Method to display information
    public void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Hobbies: " + hobbies);
    }

    // Main method (entry point of the program)
    public static void main(String[] args) {
        // Creating an instance of MyClass
        MyClass myObj = new MyClass("John", 30);

        // Adding hobbies
        myObj.addHobby("Reading");
        myObj.addHobby("Gardening");

        // Displaying information
        myObj.displayInfo();
    }
}
```

In the Example File:

1. **Package Declaration** - This line declares that the `MyClass` class is part of the `com.example package`. Packages are used to organise classes and avoid naming conflicts.
2. **Import Statements** - These statements import classes from the java.util package that are used in the `MyClass` class. In this case, `ArrayList` and `List` are imported to work with lists of hobbies.
3. **Class Declaration** - This line declares a class named `MyClass`. The public keyword indicates that the class is accessible from other classes.
4. **Variables** - These lines declare three instance variables (name, age, and hobbies). These variables are private, meaning they can only be accessed within the `MyClass` class.
5. **Constructor** - This is a constructor method that initialises the name, age, and hobbies variables when a new instance of `MyClass` is created. It takes two parameters (name and age) and initialises the hobbies list.
6. **Methods** - These are methods that define the behaviour of the `MyClass` class. The `addHobby` method adds a new hobby to the hobbies list, and the `displayInfo` method prints the name, age, and hobbies of the object.

QUIZ

What is the correct character for terminating a statement in Java?

A. A colon (:)

B. An end-of-line character

C. A tab character

D. A semicolon (;)

QUIZ ANSWER

What is the correct character for terminating a statement in Java?

A. A colon (:)

B. An end-of-line character

C. A tab character

D. A semicolon (;)

Unlike with some other programming languages, the proper way to terminate a line of code is with a semicolon (;), making D the only correct answer.

Package Statement

```
// package statement is the first statement in your class
package certification;
class Course{
  // rest of your code goes here
  }
```

Package Statement

```
// if you place the package statement after the class definition,
code won't compile

class Course{

  // rest of your code goes here

  }

package certification;
```

Package Statement

```
// class can't have multiple package statements
package cert;
package exams;
class Course{
  // rest of your code goes here
  }
```

QUIZ

Which of the following lines of code is not allowed as the first line of a Java class file?

A. import widget.*;

B. // Widget Manager

C. package sprockets;

D. int facilityNumber;

QUIZ ANSWER

Which of the following lines of code is not allowed as the first line of a Java class file?

A. import widget.*;

B. // Widget Manager

C. package sprockets;

D. int facilityNumber;

A class can start with a comment, an optional package statement, or an import statement if there is no package statement. It cannot start with a variable definition, making Option D the correct answer.

QUIZ

Which one of the following statements is true about using packages to organize your code in Java?

A. Every class is required to include a package declaration.

B. To create a new package, you need to add a package.init file to the directory.

C. Packages allow you to limit access to classes, methods, or data from classes outside the package.

D. It is not possible to restrict access to objects and methods within a package.

QUIZ ANSWER

Which one of the following statements is true about using packages to organize your code in Java?

A. Every class is required to include a package declaration.

B. To create a new package, you need to add a package.init file to the directory.

C. Packages allow you to limit access to classes, methods, or data from classes outside the package.

D. It is not possible to restrict access to objects and methods within a package.

Classes may be defined without a package declaration and are placed in the default package, so Option A is incorrect. Option B is a completely false statement as no such file is required in Java. Option C is correct as it is one of the primary reasons for organizing your application into packages. Option D is incorrect as package-private allows access to methods and variables to be limited to those classes within the same package.

Import Statement

1. Classes and interfaces in the same package can use each other without prefixing their names with the package name.
2. But to use a class or an interface from another package, you must use its fully qualified name.
3. Import statements in Java are used to make classes and interfaces from other packages available to your current source file.
4. When you import a class or interface, you're telling the compiler where to find that class or interface when it's referenced in your code.
5. Import statements help keep your code clean and readable by reducing the need for fully qualified class names.
6. They also help prevent naming conflicts when multiple classes with the same name exist in different packages.

Import Statement

1. For example, if you want to use the **`ArrayList`** class from the **`java.util package`**, you need to import it at the beginning of your Java file like this:

   import java.util.ArrayList;

2. After importing **`ArrayList`**, you can use it in your code without specifying the full package name every time: ArrayList<Integer> list = new ArrayList<>();

3. You can also import the whole content of a package using the asterisk symbol: eg

   `import java.util.*;`

Import Statement

1. Automatic Import: Classes in the java.lang package are automatically imported into every Java source file. This means that you don't need to explicitly import classes like String, System, and others from this package using an import statement. They are readily available for use in any Java program.
2. Classes in the java.lang package are automatically imported, so you don't need to include an import statement for them explicitly. This simplifies code and makes it more concise, as you can directly use classes from this package without additional import statements.
3. Java documentation:

   https://docs.oracle.com/javase/8/docs/api/java/lang/package-summary.html

QUIZ

Which statement about import statements is true?

A. The class will not compile if it contains unused import statements.

B. Unused import statements can be removed from the class without causing a class to become unable to be compiled.

C. The class will not compile if a duplicate import statement is present.

D. If a class contains an import statement for a class used in the program that cannot be found, it can still compile.

QUIZ ANSWER

Which statement about import statements is true?

A. The class will not compile if it contains unused import statements.

B. Unused import statements can be removed from the class without causing a class to become unable to be compiled.

C. The class will not compile if a duplicate import statement is present.

D. If a class contains an import statement for a class used in the program that cannot be found, it can still compile

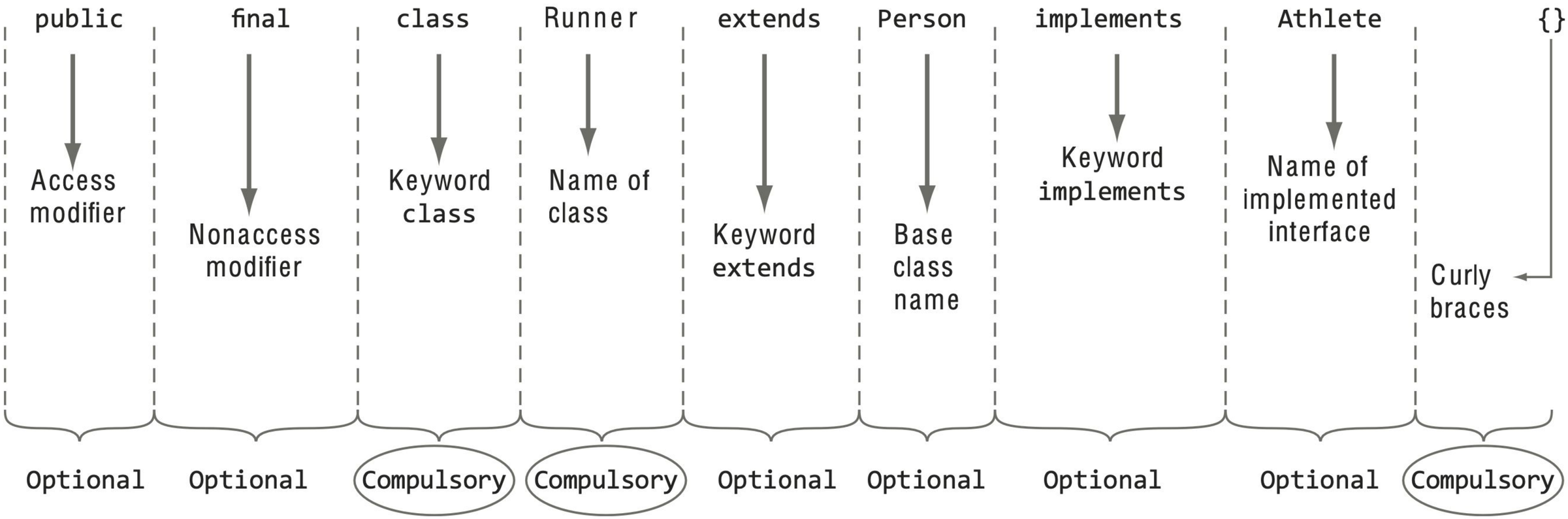
A class will compile if it has unused or redundant import statements, making Option A and C incorrect. Option D is also incorrect as the compiler must be able to locate the class of the import statement. The correct answer is Option B. Removing unused import statements does not cause a class to become unable to be compiled.

Class Declaration

1. A class is a blueprint for creating objects.
2. It defines the properties (variables) and behaviours (methods) that objects of the class will have.
3. To declare a class in Java, you use the class keyword followed by the class name and the class body enclosed in curly braces `{ }`.

Class Declaration

```
public final class Runner extends Person implements Athlete {}
```



Class Definition

1. A class is a design used to specify the attributes and behaviour of an object.
2. The attributes of an object are implemented using variables, and the behaviour is implemented using methods.
3. For example, consider a class as being like the design or specification of a mobile phone, and a mobile phone as being an object of that design.
4. The same design can be used to create multiple mobile phones
5. A class is a design from which an object can be created.
6. Variables can be put anywhere in the class, for example variable weight is defined after the constructor. This is not recommended because it makes the code difficult to read, better practice to put variables after class declaration.

```
class Phone {
   String model;
   String company;
   Phone (String model) {
   this.model = model;
   }
   double weight;
   void makeCall (String number) {
      // code
   }
   void receiveCall () {
      // code
   }
}
```

Points to remember

1. A class name starts with the keyword class.
2. Java is cAsE-sEnSiTivE.
3. The best practice for class names is to use capital letter for the first letter of the class eg (MotorBike, MobilePhone etc), this called PascalCase
4. Class names cannot start with symbols
5. The state of a class is defined using attributes or instance variables.
6. The behaviour is defined using methods, which may include method parameters.
7. A class definition may also include comments and constructors.