Homework 1

Q1) (D)

An entry point is where the first instructions of a program are executed. Execution of a program can begin at a named point, either with a conventional name defined by the programming language or operating system or at a caller-specified name. In JVM languages such as Java the entry point, defined by the programming language itself, is a method called '**main**'.

- The **main** method has to strictly follow its syntax; otherwise **JVM** will not be able to locate it and the program will not run.

- The method must be declared **public** and **static**, it must not return any value, and it must accept a String array as a parameter. The method declaration must look like the following:

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

- We can declare the **main** method as **final**. There is no restriction about main method being final.

- The method must be **static** because then it can be invoked by the runtime engine **without** having to instantiate any objects.

- The rationale behind enforcing main as public is more to do with language specification rather than whether something could be achieved or not. JNI, launching java application will never have any issue in calling a private main but this is like a jailbreak (like another, where reflection API let you access private method) and definitely not in spirit of java language specification. So to keeping it obvious and non-confusing can be the reason why JLS enforces this.

Q2) (A)

The diagram given does not demonstrate platform independence of java in anyway.

- A class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects. So it would be accurate to say that given class diagram demonstrates object-oriented design

- Unless declared with private keyword, class attributes are inherited in java.

- For Gold class to inherit luster attribute from silver class,gold class must be a subclass of silver class, but they're just sibling classes derived from the same parent class hence the luster attribute is not inherited.

Q3) (C)

Java compiled bytecode file extension name is '.class'. Code files are named with '.java' extension.

- '.dll' extension is a dynamic link library file format and used in Microsoft Windows operating systems.

- There is no file extension name called '.bytecode' in java.

Q4) (B)

Both imported **java.util** and **java.sql** packages contain a class named Date. Line 4 does not use fully qualified name for the class, as in line 5, to explicitly specify which class is going to be initialized causes ambiguity.

- Java compiler does not inhibit importing packages which contains classes of the same name.

- Using a fully qualified name (as **java.util.Date**) does not throw and exception because we are explicitly telling to compiler which package the class of the same name resides in to circumvent a situation where ambiguity arise.

Q5) (A)

Object oriented programming is based upon objects, which can contain data in the form of fields, and code in the form of procedure which acts on related data in objects.

- In java , with casting, it is possible to instruct the compiler to change the existing type of an object reference to another type.

- Objects holding data is one of the primal key features of object oriented programming languages.

- Objects performs various actions on data via the procedures they contain. This is also one of the primal key features of object oriented programming languages. In fact, setting aside some functional programming features added to object oriented languages, methods in object oriented languages cannot exists out of context of a class, which is to say they are not first-class citizens.

Q6) (D)

Local variables in java are variables that’s declared within the body of a method. They’re scoped to that method they’re declared in, Which is to say they’re only usable in that method.

- Variables declared inside interfaces are implicitly public static final variables, which is to say constants. Because they’re public they’re accessible to world, and because they’re static, there is no need any implementation of the interface to access the variable.

- Member variables are in the class level scope. They can be directly accessed anywhere in the class including.

Q7) (B)

Since the “**java.lang**” is the package which contains most of the required components, it is imported implicitly. Everything in java.lang is imported by default. For example, **java.lang.System** or **java.lang.String**.

- **java.util** is a built-in java package, but not implicitly imported. Other packages are not in the built-in java packages.

Q8) (C)

There are three types of comments in java; single-line comments, multi-line comments and documentation comments. Hashtag is not a commenting character in java.

- **//comment** syntax is for single-line comments.

- **/* this is a multi-line comment */** syntax for multi-line comments (also accepting additional * characters).

- **/** documentation comment */** syntax for documentation comments.

Q9) (D)

A .java file may contain any number of top-level classes and interfaces but only one public top-level class or interface per file is permitted. Also If there are no public top-level types then the file can be named anything.

Q10) (B)

When we consider ‘**main**’ method is an instance method, we need to think ‘accessibility at **P1’** in the context of an instance.

- Since instances can access class variables, both static variables are accessible in **‘main’** method.

- Instance can access it’s both variables (instance variables).

- Local variable is accessible also since it is in the so-called **P1** block**.**

Q11) (B)

Deleting unused imports does not cause a class to become unable to be compiled. If duplicate import statements exits, one of them is seen as an unused import statement.

- Compiler should be able to locate imported classes used in the program or else it would not compile.

Q12) (A)

The code in the example does not compile because instances are not accessible from the static context. Static context is the context when the class is loaded into JVM. Instance variables are created when an instance of a class is created, that is to say, we are trying to access an instance variable without even an instance exists in the first place.

Q13) (D)

- ‘**java**’ command can only execute compiled .class files.
- Java programming language is an object oriented programming language. It was one of the five primary goals of the principles in the creation of the language.
- ‘**javac**’ command compiles java code into bytecode, JVM converts bytecode into native machine code.

Q14) (D)

Java expects package statement as the first line of a java class file if the class belongs to a package and then import statements if there is any. If the package is the default package, imports are allowed as the first statement. Comments are not evaluated so it can start with a statement too. File does not compile if the first statement is any other statement.

Q15) (C)

Packages are used in java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier.

- Classes that is not in a package are not required to include package declaration. Classes with no package declarations are implicitly part of an '**unnamed package**', often also called '**default package**'.

- There is such .init file in java.

- The access level of a default modifier is only within the package. It cannot be accessed from outside the package.

Q16) (B)

'**javac**' command requires full filename with .java file extension, '**java**' command does not. Since the current directory is /user/home and the file is in the home directory, just specifying the filename using relative path is enough.

Q17) (D)

Encapsulation refers to the bundling of data with the methods that operate on that data, or the restricting of direct access to some of an object's components. ENcapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them. Publicly accessible methods are generally provided in the class (so-called **getters** and **setters**) to access the values, and other client classes call these methods to retrieve and modify the values within the object.

- Inheritance is the mechanism in which one class acquires the property of another class.
- Object orientation is a software engineering concept, in which concepts are represented as objects.
- Platform independence is a term that describes a technology that you can use to implement things on one machine and use them on another machine without (or with minimal) changes.

Q18) (D)

Variables can be declared inside a conditional statement. However cannot be accessed in a different scope. When declared in an if statement for example, variable is only in scope until the end of '**}**' (end of the statement). Outside of that scope, variable does not exist.

Q19) (A)

When we wish to run .class file on any other platform, we can do so. After the compilation, the bytecode generated is now run by the Java Virtual Machine and not the processor in consideration. This essentially means that we only need to have basic java installation on any platform that we want to run our code on.

- Since the file consist of JVM machine code, it’s not in a human readable content. Thus cannot be easily read and edited in a standard text editor.

- After compilation, .java file is not needed to distribute the corresponding bytecode.

Q20) (D)

Semicolon is the statement separator-terminator used in java programming language. Many other languages such as C and C++ use semicolon as statement operator too.

- EOL (end of line) is also used in many other languages as statement separator-terminator. One example is Python programming language.

- Colon (:) used in languages such as C and C++ to separate expressions.

Q21) (C)

Variable ‘**today**’ is a local variable and has a local scope to the method. Variable ‘**tomorrow**’ declared in method shadows instance variable ‘**tomorrow**’, but because it is accessed through fully-qualified name using instance reference it is evaluated to instance’s ‘**tomorrow**’ variable, not the local one. Static variable ‘**yesterday**’ is also accessed through instance reference which is discouraged and not recommended but works.

Q22) (C)

The only line that does not has a syntax error is the line 3.

- Class keyword is missing when defining a class leads to error.
- Using two different types for a variable is not a valid syntax.
- Private keyword should be at the beginning of a line instead of after the return type.

Q23) (D)

Platform independence is the feature of java that allows a Java class to be run on a wide variety of devices. Platform independence and other terms are mentioned in the preceding answers.

Q24) (A)

Java virtual machines execute Java bytecode instructions.Since this bytecode is a higher level representation than traditional object code, it is possible to decompile it back to Java source. Many such decompilers have been developed and the conventional wisdom is that decompiling Java bytecode is relatively simple.

- JVM Itself is not platform independent but supports platform independence. Java Virtual Machine provides the environment to execute the java file. On every platform that has a JVM that is built for it, you can run your java application.

- Java does memory management automatically. Java uses an automatic memory management system called a garbage collector. Thus, we are not required to implement memory management logic in our application.

- In JVM there are two main components that perform all the jobs to convert the bytecode to native machine code (Classloader and Execution Engine).

Q25)

A static variable is common to all the instances (or objects) of the class because it is a class level variable. In other words you can say that only a single copy of static variable is created and shared among all the instances of the class. Memory allocation for such variables only happens once when the class is loaded in the memory. Static variables have the longest scope; they are created when the class is loaded, and they survive as long as the class stays loaded in the Java Virtual Machine (JVM).

- There is no package variables in java.
- Instance variables have the scope of the instance they belong. They are instance specific and are not shared among instances.
- In Java, the scope of a local variable is the body of the method in which it is declared.

Q26) (C)

At first, packages appear to be hierarchical, but they are not. Importing **movie.director.*** imports all of the types in the **movie.director** package, but it does not import for example, **movie.director.recurring** package or any other **movie.director.xxxx** package

- **movie.directors** is used instead of **movie.director**
- Wildcard is used on **movie.director** package not directly on **movie** package.

Q27) (D)

Correct order of statements in order includes: package statement if file is not in the default package, import statements if there is any and then the class declaration.

Q28) (D)

java.lang package is implicitly imported by java so both of the imports can be removed. One of the **star** package imports can be safely removed too since both of this imports include **Blackhole** type.

Q29) (C)

theInput[2] corresponds to third argument given to the program. Making only C option the correct answer. Noting double quotes, in A option third argument is 'White-tailed deer'. In D option there is only two arguments so trying to access third argument gives `ArrayIndexOutOfBoundsException` runtime exception.

Q30) (B)

It's the javac command that compiles a .java file into a .class file. The java command runs the product of javac command which is the compiled bytecode on the JVM.

Q31) (B)

Method overloading is one of the features of Java programming language.

- Java is not a procedural programming language, it is object-oriented.
- Java does not allow operator overloading like in such languages as C and C++
- The java does not give direct access. Java program runs in Java Virtual Machine that controls the memory management. This means that program cannot directly access memory. Java does not have pointers like in C language to access memory.

Q32) (D)

There no null return type in java and since the function return a value return type cannot be void. Option C is wrong because there is an int keyword instead of an class keyword. Option D is the answer that would allow the code to be compiled without any issue.

Q33) (A)

Even though class has a constructor, given parameter is ignored and re-assigned to the value of 4, so **end** variable has always the value of 4. '**distance'** is printed without change as it is given to the method.

- First System.out.print always prints 2 making B and C incorrect.
- D is incorrect, the code compiles without any issue.

Q34) (D)

Code reusing is one of the most important benefits of inheritance. Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototype.

- Option A is actually a false statement. Inheritance doesn't require related classes through inheritance to be in the same package.
- Option B and C are statements against inheritance. These are some of the disadvantages of inheritance.

Q35) (A)

Any text between // and the end of the line is ignored by Java (will not be executed). So as long as line starts with two slashes, having more than two slashes does not affect single-line comment syntax.

Q36) (B)

A valid entry-point class method must do the following; be named **main**, be **public** and **static**, have a **void** return type, have a single argument with an array of type **String**. From Java 5 onward, the main method's argument type may be a String varargs instead of a string array. main can optionally throw exceptions, and its parameter can be named anything, but conventionally it is **args**. Making the main method final is optional.

Q37) (B)

- In java, everything is packed within classes so having a variable outside of a class is not valid.
- Local variables can not have access modifiers as the scope of a local variable is the body of the method in which it is declared.

Q38) (B)

- A java class file must have a class declaration.
- Package statement and imports are optional.
- A class can be package-private.

Q39) (D)

- Java source files' filename extension is **.java**.
- Compiled bytecode filename is **.class**.

Q40) (C)

The code does not compile because of line 6. Both complex package and java.lang package which is imported by default have a Math class and that makes an ambiguity arise. We should explicitly specify the package in which the Math class we want to use resides.

Q41) (A)

Although **dog.puppy** package is included, that does not include the packages inside **dog.puppy** package as well, so we need to use the fully qualified package name for classes in the **dog.puppy.female** package. It's possible to use packaged classes or interfaces without using the import statement but, by using their fully qualified name.

- Option B and C are included in the import statements and since the java.lang package is automatically imported, classes in these packages can be accessed without fully qualified package name.

Q42) (B)

Object orientation is the technique of structuring programming data as a unit consisting of attributes, with actions defined on the unit.

- Encapsulation is the technique to prevent unwanted access and manipulation of the fields or the state of a class.
- Polymorphism enables instances of java classes to exhibit multiple behaviors for the same method calls.

Q43) (A)

All of the import statement are required here as classes from all three of them are used in the class without fully qualified name. We cannot discard any import or the code will not compile.

Q44) (C)

The code does not compile because an instance variable is being tried to be accessed from a static method. Instance variables cannot be accessed from static context. If it was a static variable, code would print 'true false' to the console.

Q45) (D)

The code compiles and run without an issue and prints 20. '**feet**' and '**tracks**' variables locally scoped so they get evaluated. Class variable '**wheels**' is accessed through instance and have the value of 1. At first, it seems that local '**tracks**' variable shadows instance variable '**tracks**' but since the local variables scope is static, we couldn't access instance variable even if there wasn't a local variable named '**tracks**'. Finally, we should note that accessing class variables through instances is not a good practice and strongly discouraged.

Q46) (D)

Compiled and executed code print “**purple**” to the console. Instance variable color which has the value “**red**” is ignored because of local scope, and because color is reassigned in **printColor** method “blue” is not printed.

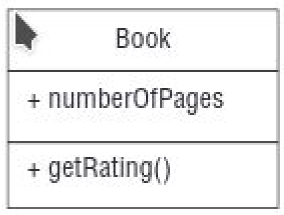
Q47) (C)

- ‘**javac**’ command takes .java files and compiles them into JVM executable .class files, so second statement is true.
- ‘java’ command uses period (‘**.**’) to separate packages not a slash (‘/’).

Q48) (D)

The code compiles but throw an error at runtime because there no proper entry point for the program to start. Method signature is not what it should be. Examining the code we see that the program argument are missing from the main method. Proper entry point may be; **“public static void main(String...args)”** .

Q49) (C)



Above diagram describes a class named ‘**book**’, having a variable named ‘**numberOfPages**’ and a method named ‘**getRating**’. Option C is the only option that properly defines given variable and method.

Q50) (C)

JVM requires a properly defined entry point method to execute the application.
- We can write an application that stuck in an endless loop.
- If the platform doesn’t have a compatible JVM which is built for it, written application cannot run on it since java applications runs on JVM.
- We can determine only which objects are eligible to be garbage collected. It can’t be determined when a particular object will be garbage collected. A user can’t control or determine the execution of a garbage collector. Its controlled by the JVM.