Ans 80) Manipulators are helping functions that can modify the input/output stream. It does not mean that we change the value of a variable, it only modifies the I/O stream using insertion (<<) and extraction (>>) operators.

* Manipulators are special functions that can be included in the I/O statement to alter the format parameters of a stream.
* Manipulators are operators that are used to format the data display.
* To access manipulators, the file iomanip.h should be included in the program.

For example, if we want to print the hexadecimal value of 100 then we can print it as:

cout<<setbase(16)<<100

Ans 81) A **token** is the smallest element of a program that is meaningful to the compiler. Tokens can be classified as follows:

1. Keywords
2. Identifiers
3. Constants
4. Special Symbols
5. Operators

1.Keywords : Keywords are pre-defined or reserved words in a programming language. Each keyword is meant to perform a specific function in a program. Since keywords are referred names for a compiler, they can’t be used as variable names because by doing so, we are trying to assign a new meaning to the keyword which is not allowed.

2.Identifier : Identifiers are used as the general terminology for naming of variables, functions and arrays. These are user-defined names consisting of an arbitrarily long sequence of letters and digits with either a letter or the underscore(\_) as a first character. Identifier names must differ in spelling and case from any keywords. You cannot use keywords as identifiers; they are reserved for special use. Once declared, you can use the identifier in later program statements to refer to the associated value.

3.Constants : Constants are also like normal variables. But, the only difference is, their values can not be modified by the program once they are defined. Constants refer to fixed values. They are also called as literals.

Constants may belong to any of the data type.

Final datatype variable name;

4.Operators: Java provides many types of operators which can be used according to the need. They are classified based on the functionality they provide. Some of the types are

 Unary Operator,

 Arithmetic Operator,

 Shift Operator,

 Relational Operator,

 Bitwise Operator,

 Logical Operator,

5. Special symbols : The following special symbols are used in Java having some special meaning and thus, cannot be used for some other purpose.

[] () {}, ; \* =

* Brackets[]: Opening and closing brackets are used as array element reference. These indicate single and multidimensional subscripts.
* Parentheses():These special symbols are used to indicate function calls and function parameters.
* Braces{}: These opening and ending curly braces marks the start and end of a block of code containing more than one executable statement.
* comma (, ): It is used to separate more than one statements like for separating parameters in function calls.
* semi colon : It is an operator that essentially invokes something called an initialization list.
* asterick (\*): It is used to create pointer variable.
* assignment operator: It is used to assign values.

A 82)Structured Programming Approach, as the word suggests, can be defined as a programming approach in which the program is made as a single structure. It means that the code will execute the instruction by instruction one after the other. It doesn’t support the possibility of jumping from one instruction to some other with the help of any statement like GOTO, etc. Therefore, the instructions in this approach will be executed in a serial and structured manner. The languages that support Structured programming approach are:

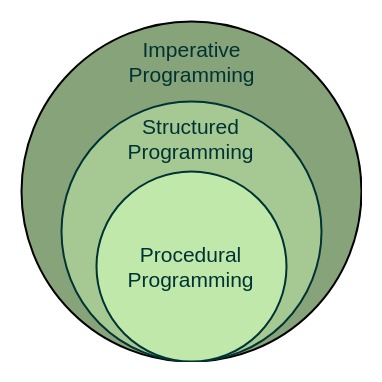
* C
* C++
* Java

Advantages of Structured Programming Approach:

1. Easier to read and understand
2. User Friendly
3. Easier to Maintain
4. Mainly problem based instead of being machine based
5. Development is easier as it requires less effort and time
6. Easier to Debug
7. Machine-Independent, mostly.

Disadvantages of Structured Programming Approach:

1. Since it is Machine-Independent, So it takes time to convert into machine code.
2. The converted machine code is not the same as for assembly language.
3. The program depends upon changeable factors like data-types. Therefore it needs to be updated with the need on the go.
4. Usually the development in this approach takes longer time as it is language-dependent. Whereas in the case of assembly language, the development takes lesser time as it is fixed for the machine.



A 83) As we know that abstraction refers to hiding the internal implementation of the feature and only showing the functionality to the users. i.e. what it works (showing), how it works (hiding).

Consider using abstract classes if any of these statements apply to your situation:

* In the java application, there are some related classes that need to share some lines of code then you can put these lines of code within the abstract class and this abstract class should be extended by all these related classes.
* You can define the non-static or non-final field(s) in the abstract class so that via a method you can access and modify the state of the object to which they belong.
* You can expect that the classes that extend an abstract class have many common methods or fields, or require access modifiers other than public (such as protected and private).

Consider using interfaces if any of these statements apply to your situation:

* It is a total abstraction, all methods declared within an interface must be implemented by the class(es) that implements this interface.
* A class can implement more than one interface. It is called multiple inheritances.
* You want to specify the behaviour of a particular data type but are not concerned about who implements its behaviour.

Abstract classes should be used primarily for objects that are closely related, whereas interfaces are best suited for **providing common functionality to unrelated classes**. If you are designing small, concise bits of functionality, use interfaces. If you are designing large functional units, use an abstract class.

A 84) There are various scenarios where we can use private constructors. The major ones are

1. Internal Constructor chaining
2. Singleton class design pattern

What is a Singleton class?

As the name implies, a class is said to be singleton if it limits the number of objects of that class to one.

We can’t have more than a single object for such classes.

Singleton classes are employed extensively in concepts like Networking and Database Connectivity.

Design Pattern of Singleton classes:

The constructor of singleton class would be private so there must be another way to get the instance of that class. This problem is resolved using a class member instance and a factory method to return the class member.

A 85) No, we cannot override private or static methods in Java.

Private methods in Java are not visible to any other class which limits their scope to the class in which they are declared.

In Java, methods declared as private can never be overridden, they are in-fact bounded during compile time.

A 86) Inheritance is a substantial rule of any Object-Oriented Programming (OOP) language but still, there are ways to prevent method overriding in child classes which are as follows:

1.Using static method

This is the first way of preventing method overriding in the child class. If you make any method static then it becomes a class method and not an object method and hence it is not allowed to be overridden as they are resolved at compilation time and overridden methods are resolved at runtime.

2. Using private access modifier  
   
Making any method private reduces the scope of that method to class only which means absolutely no one outside the class can reference that method.

3. Using default access modifier

This can only be used when the method overriding is allowed within the same package but not outside the package. Default modifier allows the method to be visible only within the package so any child class outside the same package can never override it.

4. Using the final keyword method

The final way of preventing overriding is byusing the final keyword in your method. The final keyword puts a stop to being an inheritance. Hence, if a method is made final it will be considered final implementation and no other class can override the behaviour.

A 87) For some user-defined types or type members who have a default access level, you can’t specify them explicitly at all. For example interface and enum members are always public and no access modifiers are allowed. If you are trying to add any access modifiers explicitly, it causes a **compile-time error**.

A 88)

A 89) If a class has multiple methods having same name but parameters of the method should be different is known as Method Overloading.

* If we have to perform only one operation, having same name of the methods increases the readability of the program.
* Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you to understand the behaviour of the method because its name differs.

Method overloading in java is based on the number and type of the parameters passed as an argument to the methods. We can not define more than one method with the same name, Order, and type of the arguments. It would be a compiler error. The compiler does not consider the return type while differentiating the overloaded method. But you cannot declare two methods with the same signature and different return types. It will throw a compile-time error. If both methods have the same parameter types, but different return types, then it is not possible.

Parameters should be different means   
1. Type of parameter should be different.

void add(int a, int b){

        System.out.println(“sum =”+(a+b));

    void  add(double a, double b){

        System.out.println(“sum=”+(a+b));

2.Number of parameter should be different.

void add(int a, int b){

        System.out.println(“sum =”+(a+b));

      void  add(int a, int b,int c){

        System.out.println(“sum=”+(a+b+c));

A 90) Yes, we can declare an abstract class with no abstract methods in Java.

* An abstract class means that hiding the implementation and showing the function definition to the user.
* An abstract class having both abstract methods and non-abstract methods.
* For an abstract class, we are not able to create an object directly. But Indirectly we can create an object using the subclass object.
* A Java abstract class can have instance methods that implement a default behavior.
* An abstract class can extend only one class or one abstract class at a time.
* Declaring a class as abstract with no abstract methods means that we don't allow it to be instantiated on its own.
* An abstract class used in Java signifies that we can't create an object of the class directly.

A 91)Default: When no access modifier is specified for a class, method, or data member – It is said to be having the default access modifier by default.

* + The data members, class or methods which are not declared using any access modifiers i.e. having default access modifier are accessible only within the same package.

The fields in an interface are implicitly public static final and the methods in an interface are by default public.

Variables and methods can be declared without any modifiers, as in the following examples -

String version = "1.5.1";

boolean processOrder() {

   return true;

}

A 92) **No**,Function Overriding can only be defined in different class because it treated as Base and Derived or parent-child relationship.

A 93) In java to write a overloaded method you have to change its signature.And a method signature depends on the following things -

* method name
* number of method parameter
* types of method parameter

So based on the above facts these are the valid overloaded methods

So no, return type does not matter if the function has different return type and same signature it will show a compile time error.

A 94) **Constructor** is always called by its class name in a class itself. A constructor is used to initialize an object not to build the object. As we all know abstract classes also do have a constructor. So if we do not define any constructor inside the abstract class then JVM (Java Virtual Machine) will give a default constructor to the abstract class.

* If you define your own constructor without arguments inside an abstract class but forget to call your own constructor inside its derived class constructor then JVM will call the constructor by default.
* So if you define your single or multi-argument constructor inside the abstract class then make sure to call the constructor inside the derived class constructor with the super keyword.

A 95)