Q2) List and explain salient features of java
(8M)

→ ✱ Salient features of Java:-

① Platform Independence:- Java programs can run on any device or operating system that supports Java Virtual Machine (JVM), making it highly portable.

Example:- A java program developed on Windows machine can run seamlessly on Linux or MacOS machines without any modifications, demonstrating platform independence.

② Object - oriented:- Java is purely object - oriented, allowing modular and reusable code through classes and objects.

Example:- In java you create a class called 'Account' with methods like 'deposit', 'withdraw', and 'getbalance' which can be reused for different types of accounts such as savings acc. or checking acc.

③ Robustness :- Java provides features like strong memory management, exception handling and type checking, enhancing robustness in software developement.

Example:- Java's exception

handlling mechanism helps develo to handle runtime errors ~~to~~ gracefully.

~~(4) Security:- Java's security model restricts untrusted code from accessing system resour directly.~~

(4) Security:- Java's security feature include bytecode verifier, runtime security checks and a security manager, ensuring secure execution of code

(5) Multithreading:- Java supports concurrent execution of multiple threads, enabling efficient utilization of resources and better performance in multitasking environments.

Q8 Explain use of keywords super and this.

Q9 → (1) Super:-

- The 'super' keyword in Java is used to refer immediate parent clas object.
- It is primarily used to ~~refer~~ access methods, variables and constructors of ~~subcl~~ superclass within the subclass.
- It is particularly useful when subclass overrides the supercal and needs to call overridden method from within the subcl
- 'super()' is also used to invok the superclass constructor from subclass constructor.

Example:-

```
class Parent {
    int num = 10;
      void disp() {
         s.o.p.("Parent"
      }
  }
class Child extends Parent {
      int num = 20;
        ~~s.o.p~~(super.displ
        void disp () {
        } s.o.p ("Chold"
    void print () {
        S.o.p ("super.num");
    }
}
```

② this:-

- The 'this' keyword in Java is reference to the current instance of the class.
- 'this' is used to access instance methods, instance variables and instance constructors of current class.
- 'this' is commonly used to resolve the ambiguity between instance variables and method parameters when they have same name.
- It can also be used to invoke one constructor from another constructor within same class.

Example:-

```
Class Myclass: {
    int num;
    MyClass (int num){
        this.num= num;
    }
    void disp () {
        S.o.p (this.num);
    }
}
```

Q3) Explain how memory is allocated to objects in Java?

→ ① Object Creation:-
When object is created using 'new' keyword, the ~~ob~~ memory is allocated from heap memory area. Size of memory allocated depends on Instance variables and overheads associated with object.

② Instance variables allocation:-
Memory is allocated to all instance variables of object. Each variable occupies specific amount of memor based on its datatypes.

③ Method Allocation:- Methods of the objects are ~~stored~~ loaded into memory from the class definition but aren't store within objects memory.

④ Reference Allocation:- If the object contains reference variables, the memory is allocated to hold the references ~~of~~ to other objects or primitive types. The size of refere depends on the architecture of system (usually 4 or 8 byte).

⑤ Heap Space Management:- Java's garbage collector periodically checks for the objects that are no longer in use. It reclaims the memory occupied by those objects and making it available for new object allocations.

Q.4] Discuss behaviour of Static members in Java.

→ In Java, static members offer distinct behaviour :-

① Static variables :-
A.k.a class variables, they are shared across all class instances. Initialized once during class loading, they persist throughout the program execution.

② Static Methods :-
Belong to class rather than an specific instances. They're directly callable using class name, without object instantiation.

③ Static blocks :-
Executed during class Loading, before any instance creation or static methods invocation.

④ Static Import :-

Q5) Describe components of JVM.

→ There are several essential components of JVM :-

① Class Loader
- Responsible for loading class files into memory dynamically during run-time. The class loader subsystem consists of three main components :-

ⓐ Bootstrap C.L. :-
Loads Core Java classes required by JVM itself.

ⓑ Extension C.L. :-
Loads classes from extension directories.

ⓒ Application C.L. :-
Loads classes from classpath or application specific locations.

② Runtime Data Area :-
Memory Areas used by JVM to execute Java Programs :-

ⓐ Method Area :-
Stores class-level structure such as method bytecode, field data, runtime constants.

(b) Heap:- Memory space used for dynamic memory allocation for objects and array.

(c) Java Stack:- Each thread has its own java stack, which holds the method invocation frame containing local variables, stack operands and other relevant data's.

(3) Execution Engine:-

Responsible for executing bytecode

It includes:-

(a) Interpreter:- Reads bytecode ~~information~~ instructions and executed them one by one. Provides platform independence but can be slow.

(b) Just-in-Time (JIT) Compiler:-

Compiles frequently executed bytecode into native machine bytecode for improved performance.

(c) Garbage Collector:- Manages memory by reclaiming the memory occupied by unreachable objects to prevent memory leaks.

(4) JNI (Java Native Interface):-

Allows java code to calls and be called by native appli. and libraries written in other lang. such as C & C++.

Q7) Java is platform Independent & Strongly typed.

→ Platform Independence:-

→ The Java achieves platform independence through it's 'Write Once Run Anywhere' (WORA) ~~B~~ Principle.

- Java source code is compiled into platform - Independent bytecode by Java compiler.
- This byte is then interpreted ~~into~~ by JVM on any platform that is compatible with JVM implementation.
- Since, Java prog. are executed by JVM, they can run on any ~~p~~ system which has JVM installed, regardless of underlying hardware and O.S.. This enables Java applications to be highly portable.

Strongly Typing:-

- Java is strongly type, meaning that every variable and expression in Java has its own specific ~~data~~ type which is known at compile time. This allows compiler to perform types checking to detect and prevent type related errors before run-time.
- Java enforces strict types rules, such as not allowing operations between incompatible types and, requir

Q10) How is main () method of Java written

→ The main method in Java serves as an entry point for any Java Applicati It is where the execution of Java programs begin.

Signature:-
public static void main(String [] args)

- public (Access ~~specifier~~ Modifier) :-
  - It is declared with "public" access ~~specifier~~ modifior so that it can be accessed from anywhere within program.

- Return type (void) :-
  - In Java, main method has a return type of 'void' indicating that it ~~dose~~ does not return any value.
  - In Java, the program terminates when the main() method completes its execution, so there's no need for it to return any value.

- Static Modifior (static):-
  - The main method is declared as static, allowing it to be invoked without creating any instance of the class.
  - This is because Jvm calls ~~itself~~ the main method in class, rather than in any specific instance of the class.

- Method name('main')
  - The name of the method is main. which is standard convention of ~~any~~ entry point of ~~the~~ method in Java Programs.

- Parameters:-
  The main method accepts single parameter of type Strings conventionally named 'args'.

Example:-

```
public class Hello {
    public static void main (String [] args){
        {
            S.o.p ("Hello");
        }
    }
```

Q11) Describe role of abstract ~~@~~ & defa
Q15) methods.

→ ① Abstract ~~Classes~~ Methods:

(a) Definition:-

- An abstract ~~eto~~ method is metho without an implementation (i.e. with method ~~@~~ body).
- It is blueprint for desived classes to provide their own implementation.

(b) Declaration:-

- The abstract methods are declare with 'abstract' keyword in method signature.
- The are defined in abstract class or interface.
- Abstract classes may contain bo abstract & concrete methods, where Interfaces can only contain abstra methods.

(c) Purpose:-

- Abstract method allow defining methods in superclass or interface that must be implemented by its ~~oton~~ subclasses or implementing classes.

Q12: Primitive data type values & objects.

→ (1) Primitive data type values:-

- Primitive data type such as int, double, boolean, etc. hold their values directly into the memory.
- The primitive data type values has fixed sized in memory, determined by Java language specification.
- When primitive variable is declared, memory is allocated to hold value of that data type.
- ~~Memory alloc~~ The size of memory allocated depends on the data type. For example, 'int' typically occupies 4 bytes, while double occupies 8 bytes.
- Primitive values are stored in stack memory area, which are used for storing method invocations and local variables.
- The stack memory is organize as stack data structure, with each method call creating new stack frame that contains space for local variables, including primitive data type values.
- Primitive values are directly accessed by variable names.

(2) Primitive data type Objects:-

- Object ~~variable~~ in java are instances of classes and are stored in heap memory area.
- When Object is created using new keyword, memory is allocated in heap to store objects data.
- Object is directly not stored in variable whereas the reference to the object is stored within the variable.
- The memory allocated for an object in heap includes spaces for instance variables and reference to its class.
- Objects are stored in heap because they have longer lifespan than local variables stored in stack.
- Java employs automatic garbage collection to reclaim the memory occupied by objects which are no longer in use.