≐ Interview Questions ≐

Q.1 What is Method Overloading in Java? Explain with example?

→ Method overloading means multiple methods in a class with same name with diff^n Parameters.

Ex:
```
Public class Employee {
    Public int add(int a, int b) {
        return a+b;
    }
    Public int add(int a, int b, int c) {
        return a+b+c;
    }
}
```

Q.2 What are the rules for Method overloading resolution in Java? How does Java determine which overloaded method to call?

→ • When we call an overloaded method Java Determines which version of method to execute based on the arguments provided at a time of calling. The process of selecting an appropriate overload method is known as method overloading resolution.

Rules:

① Give no. of Parameters exactly with one of overload methods

② give Proper Data types

③ Type Promotion: If there is no exact match bet^n Argument types & Parameter types Java tries to promote the arguments to larger Data types.

④ Autoboxing & Varargs: autoboxing allows Primitive Data types automatically converted into their corresponding Wrapper Class objects & Varargs allow method to accept Variable number of arguments of same type.

⑤ Inheritance & Method overriding: If a Superclass has method that is overridden in a subclass and there are overloaded methods in the subclass, java Consider the runtime type of object to determine which overloaded method to call.

Q.3 What is static keyword Explain diff^n betn static & non-static

→ Static keyword is used to Define Variable, Method, block or nested class as belongs to the class rather than to instance of class. When a member is declared as static it means it's associated with class itself rather than with any particular instance of class.

1] Static Variable: They are shared among all instance of class only one copy of static variable exists.

Ex: Public class Abhijeet {
    static int n = 0;
}

2] Static Method: These are associated with class rather than any particular instance of class. They can be called using class name without creating instance of class

Static methods	Non-static Methods
• They belongs to class rather than any instance of class	• They belongs to individual instance of class
• Called using class name directly	• access both static & non-static of class directly.
• Static Methods can't access instance variable directly.	• non-static methods can't be called without creating an instance of class
• They can access other static members, including static variables & other static methods.	• They have access to 'this' reference.

Ex:

```
Public class myClass {
    Static int n1 = 20;
    int n2 = 30;
    Static void Method1() {
        Here we can access n1
        but can't access n2 it gives compilation error
    }
    void method2() {
        Here we can access both n1 & n2
    }
    Public static void main(String[] args) {
        Method1(); → we can call static method without creating
                     instance
        but to call method2() we have to make instance
        myClass m = new myClass();
        m.method2();
```

Q.4 Can static methods be overloaded & overridden in Java? How are static variables shared across multiple instance of class.

→ In Java static methods can be overloaded but not overridden. However overriding refers to providing a new implementation of a method in a subclass that are already exists in its superclass.

Static variables in java are shared across multiple instance of class because they belong to class itself rather than to any specific instance of class. When we declare variable as static within a class there is only one copy of that variable that is shared by all instances of class.

Ex:

```
Class MyClass {
    Static int n = 0;  // Static Variable Shared across all
                          instances.
    int instanceVariable;
    Public MyClass(int instanceVariable){
        this.instanceVariable = instanceVariable;
    }
}
Public Class Main{
    Public Static void main(String[] args){
        MyClass obj1 = new MyClass(10);
        MyClass obj2 = new MyClass(20);

        obj1.n = 5;  // Accessing & Modifying Static
                        Variable through obj1
        System.out.println(obj2.n); // Accessing Static Variable
                                       through obj2
        MyClass.n = 8; // Modifying Static Variable directly
                          using class name.
        System.out.println(obj1.n); // Accessing Static Variable
                                       through obj1
```

Constructor to initialize instance variable

Create two instances

Q.5 What is the role of Static keyword in Context of Memory Management.

→ A) Class-level Memory Allocation : When a variable or method is declared as static memory for it is allocated only once regardless of how many instance of class created This memory is allocated when class loaded into memory by JVM.

B] No instance Specific Allocation: Since 'static' members are associated with the class itself rather than with instance of class they do not contribute to memory of individual objects created for that class.

C] Accessing Memory: Static memory can be accessed directly using the class name without the need to create an instance of class.

Q.6 What is significance of final keyword in Java?

→ In Java final keyword used to denote variable, method or class it can't be modified

Final Method: When we create method as final we can't override by subclass.

Final class: When class is final it means class can't be subclasses.

Q.7 Can final method be overridden in subclass? How does final keyword affect variable, methods & classes?

→ 1) Variable: If a variable is declared as final its value can't changed once its has been initialized.

2) Method: If method is final we can't perform overridden by subclass

3) Class: When class is final it means class can't be subclasses.

Q.8 What does this keyword represent in Java? How is the this keyword used in constructors and methods?

→ this keyword is a reference to current instance of class. it can be used within constructors & methods to refer current object on which method is being called.

Q.9 What is narrowing & widening conversions in Java?

→ 1) Widening: Convert value of smaller Data type to larger Datatype
Ex: int to double

2) Narrowing: Convert value of larger Data type to smaller Datatype
Ex: double to int.

Q.10 Provide examples of Narrowing & Widening Conversions b/w Primitive data types

→ 1] Widening Conversion:

```
int n1 = 10;
long n2 = n1;
```

2] Narrowing Conversion:

```
double d = 10.5;
int i = (int) d;
```

Q.11 How does Java handle Potential loss of precision during narrowing conversions?

→ 1] Using Implicit Narrowing → Ex: int to byte

```
int i = 10;
byte b = (byte) i;
```

2] Truncation → During conversions Java truncates the higher order bits of value, retaining only the lower order bits.

Ex: Float to int

```
double d = 123.487;
int i = (int) d; // Here 123 is retained & decimal part is truncated
```

3] Potential Data loss.

4] Casting: Java requires explicit casting when performing narrowing conversions to indicate or aware Potential loss.

- Java Provides Mechanism for handling Potential loss of Precision during narrowing Conversions through explicit Casting & truncation

Q.12 Explain concept of automatic widening conversion in Java.

→ Automatic widening is converting smaller Data type automatically converted to larger Data type. without the need for explicit casting

Q.13 What are implication of narrowing & widening conversions on type compatibility & data loss?

→ Narrowing & widening conversions have implications for type compatibility, data loss, compatibility / probability & safety / correctness. Widening conversions are generally safer while narrowing conversions require more attention to avoid data loss & ensure correctness.