Assignment no.- 6

1) What is method overloading in Java & explain with an example?

Method overloading → If a class has multiple methods having same name but different in parameters. it is known as Method Overloading.

```
→ Class Adder {
    Static int add (int a, int b){ return a+b; }
    static int add (int a, int b, int c) { return a+b+c; }
  }
  Class TestOverloading1 {
   public static void main (String[] args) {
    System.out.println (Adder. add (11, 11));
    System.out.println (Adder.add (11, 11, 11);
   }}
```

Output
```
22
33
```

2) What are the rules for method overloading in java? How does java determine which overloaded method to call?

3Ways to overload a method: It is done by changing:

1. Number of parameters in Two methods.
2. The datatype of the parameters of methods.
3. The order of the parameters of methods.

Java determines which method to call based on the number and types of arguments passed to the method. This process is called method resolution, or method overloading resolution.

3) What does the static keyword mean in java? Explain the difference between static and non static methods:

Static → Static keyword in java indicates that a particular member is not an instance, but rather part of a type. The static member will be shared among all instance of the class, so we will only create one instance of it.

Difference between static and non-static method

Points	Static method	Non static method
Definition	A static method is method that belongs to a class. but it does not belong to an instance of that class and this method can be called without the instance or object of that class.	Every method in java defaults to a non-Static method ~~and~~ w/o a static keyword preceding it. non-static methods can access any static method and static variable also, without using the object of the class.
Accessing members & methods	In the static method, the method can only access only Static data members and static methods of another class or the same Class but cannot access non static methods and variables.	In the non static method the method can access static data members and static methods as well as non static members and methods of another class or the same class.
Binding process	The static method uses compile time or early binding.	The non static method uses runtime or dynamic binding.
Overriding	The static method cannot be overridden because of early binding.	The non static method can be overridden because of runtime binding.
Memory allocation	In the static method, less m/m is used for execution because memory allocation happens only once because of the static keyword fixed a particular memory for that method in ram.	In the non-static method, much memory is used for execution because there memory allocation happens when the method is invoked and the memory is allocated every time when the method is called.

4) Can static methods be overloaded and overridden in Java? How are static variables shared across multiple instances of a class?

→ In Java, static ~~or~~ methods cannot be ~~overloaded~~ overridden because they belong to the class rather than the instance of the class. However they can be overloaded. Overloading means having multiple methods in the same class with the same name but different parameters.

→ Static variables in Java are shared across all instances of a class because they belong to the class itself rather than to any specific instance. When you declare a variable as static within a class, there's only one instance of that variable that is shared among all instances of the class. here's how it works;

1) Memory Allocation → When a class is overloaded into memory, space for its static variables is allocated. These variables are initialized only once, regardless of how many instances of the class are created.

2) Access → Static variables can be accessed using either the class name or an instance of the class. However, it's recommended to access them using the class name to make it clear that the variable is static and shared among all instances.

3) Changes → Any changes made to a static variable are reflected across all instances of the class. If one instance modifies the static variable, all other instances will see the updated value.

5) What role of the static keyword in the context of memory management.

~~It plays a significant role in memory management primarily in two aspects:~~

~~1) Static Variables Allocation → When you declare a variable~~

The static keyword in Java is mainly used for memory management. The static keyword in Java is used to share the same variable or method of a given class. The users can apply static keyword with variables, methods, blocks and nested classes. The static keyword belongs to the class than an instance of the class.

6) What is the significance of the final keyword in Java?
In Java, the final keyword is used to indicate that a variable, method or class cannot be modified or extended. Here are some of its characteristics:

(7)(6) <u>Final variables</u>: its value cannot be changed once initialized. This is useful for declaring constants or other values that should not be modified.

<u>Final methods</u>: When a method is declared as final, it cannot be overridden by a subclass. This is useful for methods that are part of a class's public API and should not be modified by subclasses.

<u>Final class</u>: It cannot be extended by a subclass.

7) Can a final method be overridden in a subclass? How does the final keyword affect variables, methods and classes in Java?
→ In Java, a 'final' ~~class~~ method cannot be overridden in a subclass. When a method is declared as 'final' in a superclass, it means that it cannot be overridden by any subclass. This is useful when you want to enforce that a particular method implementation remains unchanged throughout the class hierarchy. Attempting to override a final method in a subclass will result in a compilation error.

8) What does the this keyword represent in Java? How is the this keyword used in constructors and methods?
The This keyword refers to the current object in a method or constructor. The most common use of this keyword is to eliminate the confusion between class attributes and parameters with the some name (because a class attribute is shadowed by a method or constructor parameter).

Here's how its used in constructors and methods :

1) Referencing Instance Variables: Inside a method or constructor, if there's a local variable or parameter with the same name as an instance variable, usin 'this' allows you to refer to the instance variable.

```
public class MyClass {
        private int x;
        public void setInt (int x) {
                this.x = x ; // 'this' refers to the instance var. x.
        }
}
```

2) Invoking another constructor : In a constructor, 'this' can be used to call another constructor of the same class. This is often used to avoid duplicating initialization code or to provide constructors with different sets of parameters.

```
public class MyClass {
        private int x;
        public MyClass() {
                this(0); // calls another constructor with a parameter
        }
        public MyClass (int x){
                this.x = x ;
        }
}
```

3) Passing the current Object: In a method, 'this' can be used to pass ~~an~~ the current object as a parameter to other ~~parameter~~ methods. This is useful when you need to pass the current object as an argument, such as when invoking another method on the same object.

```
public class MyClass{
        private int x;
        public void printX(){
                System.out.println("Value of x:" + this.x);
        }
        public void doSomething(){
                //passes the current obj as an arg. to printX method
                this.printX();
        } }
```

9) What are narrowing and widening constructors conversions in Java? 10) Example of narrowing and widening of primitive datatypes.

Narrowing Conversion ⇒ It's also known as demotion. occurs when data is converted to a smaller data type. It may lead to loss of information. (Explicite type cast)
eg. Converting a larger integer type (like long) to smaller one (like byte or short)

```
double doublevalue = 10.5;
int value=(int) doublevalue ;   // Narrowing conversion
                                   from double to int.
```

Widening Conversion ⇒ A widening conversion, also known as widening or promotion. occurs when data is converted to a larger data type. ~~In~~ Java ~~it is~~ automatically performs widening conversions when no data loss will occur.

eg. converting a integer like float number to larger like long. or int to double.

```
int intvalue = 10;
double doublevalue = intValue ; //widening conversion from
                                   int to double
```

11) How does Java handle pontential loss of precision during narrowing conversions?
Heres how Java handles potential loss of precision during narrowing conversions:

1) Explicit Type Casting - Java mandates explicit type casting when converting from a large data type to a smaller one. This casting tells the compiler that the potential loss of precision is acceptable because the programmer has explicitly acknowledged it.

2) Truncation - During narrowing conversions, if the value being converted exceeds the range of the destination data type, the excess bits are truncated without any warning or error. This truncation can lead to unexpected results or loss of data if not handled carefully by the programmer.

3) Data loss - Since narrowing conversions involve converting a larger data type to a smaller one, there's a risk of losing information or Precision. For example converting a 'double' to an 'int' may result in truncating the fractional part of the 'double' value, leading to loss of precision.

12) Explain the concept of automatic widening conversion in Java.

In Java, automatic widening conversion is a mechanism where the compiler automaticaly converts data from a smaller data type to a larger one without requiring any explicit action from the programmer.

This conversion is done when theres no risk of losing data or pecision

Implicit conversion - This conversion is implicit, meaning the programmer doesn't need to specify any conversion operation. It's handled automatically by the Java compiler during compilation.

Preservation of data :- The conversion is performed ~~by~~ in such a way that no data is lost.

Predefined Conversion Rules - Java follows Predefined conversion rules to determine the widening conversion path. For example, when an operation involves operands of different data types. Java will automatically promote the operands to a common type using widening conversions.

Use cases: Automatic widening conversions is commonly encountered in expressions where operands have different data types.

13) What are the implications of narrowing and widening on type compatibility and data loss.

- ~~Widening Conversion~~ ~~Narrowing conversion~~

~~Type~~ Compatibility Conversion Type	Type Compatbility	Data Loss	Explicitness	Saftey.
Widening conversion	Ensures compatibility	No data loss	Implicit (no casting required)	Generally safe, automatically performed by the compiler.
Narrowing conversion	May lead to loss of compatibility	Risk of data loss	Explicit (requires casting)	Risky if not handled properly, potential for data loss.