Assignment - 6

**1)

Method overloading

If a Class has multiple methods having same name but different in Parameters it is known as method overloading these methods can be have different numbers or types of parameters. Also method overloading increase the readability of the program.

Different ways to overload the method ⇒

1) By changing number of arguments
2) By changing the data type.

1)* By changing no. of arguments ⇒

Ex.→ We have create two methods first add () methed perform addition of two numbers.

```
Public class Calculator {
Public int add (int a , int b) {
    return a + b;
}
Public class main {
Public class static Void main (String[] args)
```

```
{
    Calculator calculator = new Calculator
                                ();
    System.out.Println("sum of a and b is
                : " + calculator.add(a, b));
}
}
}
```

2)* By changing the data type ⇒

Ex We have Create two method first add () method perform addition of two double number.

```
Public Class Calculator{
    Public double add (double a, double b)
{
        return a+b;
    }
Public class main{
    Public static Void main (String[] args)
{
Calculator Calculator = new Calculator();
System.out.Print ln ("sum of a and b is
                : " + Calculator.add(a, b));
} } }
```

__/__/____

**2) Rules of method overloading resolution =>

→ The process of selecting the most appropriate overloaded function or operator is called method overloading resolution.

→ Method overloading is a feature that allow a class to have more then one method having the same name if their argument lists are different.

→ Method overloading resolution are applied during compile time and are based on the parameter provided during method invocation.

1) Exact Match => If there is an exact match b/w the parameters passed in the method and the parameter of one of the overloaded method.

2) Widening Convertion => there is a method that can accept the parameter passed through a widening Conversion. Converting a smaller size type to a longer size.

byte → short → char → int → long → float → double

3) Autoboxing => there is a method that can accept the parameter through autoboxing converting primitive type to their corresponding wrapper class or vice versa.

4) Narrowing ⇒ there is a method that can accept the parameter passed through a Narrowing Conversion converting a larger type size to Smaller type size.

**3) Static Keyword in Java ⇒

If any member in a Class is declared as static. it mean that even before the Class is initiated all the static member can be accessed and become active.

Diff° b/w static and Non static Keyword ⇒

1) Static methods are Class method while Non Static belong to an instance of the Class.

2) Static Variables are shared by all object of a class have a single instance while Non static Variable are unique to each object and have different Value for different object

3) Static can be accessed directly using the Class name followed by the member name
Ex → (ClassName . memberName) they are accessible from anywhere within the program.
Non static member are accessed using an object reference followed by the member name.
Ex → (Object Refeerence. memberName) they are specific to a perticular instance of the Class.

__/__/____

4) Static member have a global scope and can be accessed from anywhere within the program even without creating an instance of the class. Non static member have a local space and can be accessed only through an instance of the class they are not accessible without creating an object.

5) Static member can only access other static member within the same class they cannot directly access non static member. Non static member. can access both static and non static member ~~bethe~~ within the same class they have direct access to all members

**4) Static method can be overloaded but not overridden they can have different parameter while having the same name in the same class or subclass they can't be overridden because they act on the class itself not an object.

Use a static variable all instance of the same class share a single copy of the static variable.

1) Memory allocation ⇒ when a static variable is declared within a class memory for that variable is allocated only once regardless of how many instance of the class are created.

2) Scope ⇒ Static variable are scoped to the class rather than to any particular instance of the class this mean they can be accessed

using the class name itself rather than through an instance of the class.

3) Initialization ⇒ static variable are typically initialized once either when they are declared or in a static initialize block and retain their value throughout the life time of the program unless explicitly modified.

4) Access ⇒ Since static variable are associated with the class itself they can be accessed without needing an instance of the class however they can also be accessed through an instance if ~~need~~ necessary.

**5) Role of the static Keyword in the Context of memory management ⇒

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 user can apply static Keyword with variable method, block and nested classes the static Keyword belong to the class than an instance of the class. the static Keyword is used for a constant variable or a method that is the same for every instance of a class the static Keyword is a non-access modifier in Java that is applicable for the following ⇒

1) Blocks.
2) Variables.
3) Methods.
4) Classes.

__/__/____

1) Variable (also known as a class Variable)

→ Static variable gets memory only once in the class area at the time of class loading.

→ Static variable can be used to refer to the common property of all objects.

2) Method (also known as a class method)

→ A static method belongs to the class rather than the object of a class.
→ A static method can be invoked without the need for creating an instance of a class.
→ A static method can access static data member and can change the value of it.

3) block ⇒

→ It is used to initialize the static data member.
→ It is executed before the main method at the time of classloading.

4) Class ⇒

→ A class can be made static only if is a nested class. Nested static class doesn't need a reference of outer class.

**6) Significance of the final Keyword in java ⇒

The final Keyword is a non access modifier used for classes, attributes and methods which makes them non changeable (impossible to inherit or override). The final keyword is useful when you want a variable to always store the same value.

1) final Variable → To create Constant variable
2) final Method → Prevent Method overriding
3) final Classes → Prevent inheritance

1) final Variable ⇒ When a variable is declared as final its value cannot be changed once it has been initialized.

2) final method ⇒ It cannot be overridden by a subclass this is useful for method that are part of a class's Public API and should not be modified by subclasses.

3) final Classes ⇒ It cannot be extended by a subclass this is useful for classes that are intended to be used as is and should not be Modified or extended.

the final Keyword is a Useful tool for improving code quality and ensuring that certain aspects of a program (an not be modified or extended.

**7) No, the method that are declared as final cannot be overridden or hidden. the keyword 'final' serves as a non-access modifier application of classes, methods, and variables. A final class cannot be subclassed a final method cannot be overridden and a final variable cannot be reassigned once initialized.

* final keyword in Java can be used with -

i) Variables
ii) Method
iii) Classes

→ Variables with final keyword ⇒ When a final keyword is used with a variable it makes the variable constant hence once assigned the value of the varible cannot be changed.

→ Method with final keyword ⇒ the final keyword can also be used to decleared a method as final A final method cannot be overridden by a subclass.

→ Classes with final keyword ⇒ classes followed by the final keyword in Java cannot be inherited by any subclass (or child class)

__/__/____

**8) This Keyword represent in Java ⇒

→ "this" is an important Keyword in Java it helps to distinguish between local variable and variable passed in the method as parameters.

→ this Keyword is refers to the current object in a method or Constructor.

→ the most common use of the this Keyword is to eliminate the Confusion b/w Class attributes and parameter with the same name.

this Keyword used in Constructor and method ⇒

⇒ When Java Constructor is called each time an object is Created using a new() Keyword at least one Constructor (it could be the default Constructor) is invoked to assign initial value to the data members of the same class.

⇒ Using this() to invoke the current Class Constructor Using 'this' Keyword to return the current Class instance Using 'this' Keyword as the method parameter Using this Keyword to invoke the current Class method.

**9) Narrowing and Widening Conversion in Java ⇒

Narrowing ⇒ Converting a higher type size

to Smaller type Size it is also known as explicit Conversion or Casting up. It is done manually by the programmer if we do not perform Casting then the Compiler reports a Compile time error

double > float > long > int > char > short > byte.

Widening ⇒ Converting Smaller type size to larger type size it is also known as implicit Conversion or Casting down. It is done automatically.

byte < short < char < int < long < float < double.

**10) Narrowing Conversion ⇒

this occurs when we Convert a data type to one that can hold Smaller value it requires explicit casting may result in loss of precision or overflow.

Ex⇒
```
{ double doubleValue = 10.5;
  int intValue = (int) doubleValue;
  System.out.println(intValue);
}
```
output = 10.

__/__/____

Widening Conversion ⇒ this occurs when we convert a data type to one that can hold higher value without losing precision. Java Automatically Perform widening conversion.

Ex⇒

```
{ int intValue = 10;
  double doubleValue = intValue;
  System.out.PrintIn (double Value);
}
```

output = 10.0

**11) Java handle potential loss of precision during narrowing Conversion ⇒

Narrowing Conversion may result in lose of precision when Converting from a data type that can represent a larger range of value of one that can represent a Smaller type range Java handles potential loss of precision during narrowing Conversion by truncating or rounding the value being Converted When we perform a narrowing Conversion you may explicity Cast the value to the target data type.

```
double double Value = 10.5;
int intValue = (int) doubleValue;
```

output = 10

Java does not perform automatic rounding during narrowing conversion instead it simply discard the extra bits that cannot be represented in the target data type this can lead to unexpected results if the Programmer is not careful especially when dealing with floating-point number or large value.

**12) Concept of automatic widening Conversion

⇒ Automatic Widening Conversion is also known as implicit type Conversion or promotion is a feature in Java where the Java Compiler automatically converts smaller data type to larger data type when necessary this Conversion happens seamlessly without the need for explicit casting in most cases. Widening conversion are safe because they do not result in loss of precision or data.

```
int intValue = 10;

double doubleValue = intValue;

Output = 10.0
```

Automatic widening Conversion is commonly used in Java when performing arithmatic operation or assignments involving different data type it simplifies the code and make it more readable by eliminating the need for explicit type casting in many scenarios

__/__/____

**13) Narrowing and Widening Conversion in Java have implications on type compatibility and potential data loss ⇒

1) Type Compatibility ⇒

a) Widening Conversion ⇒ Widening Conversion are always safe and compatible because they involve Converting a smaller data type to a larger Value.

b) Narrowing Conversion ⇒ Narrowing Conversion can lead to loss of data or precision. they are potentially less safe than widening Conversion. Java requires explicit casting for narrowing Conversion. to indicate that the programmer is aware of the potential loss of data.

2) Data loss ⇒

a) Widening Conversion ⇒ Widening Conversion do not result in data loss Since the target data type can represent all Value of the source data type.

b) Narrowing Conversion ⇒ Narrowing Conversion Can lead to data loss or loss of precision. especially if the Value being Converting exceeds the range or precision of the target data type.