Date .....
Page .....

Assignment NO :- 6

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

Ans:- Method overloading in Java refers to the ability to define multiple methods in a class with the same name but with different parameter lists. These methods can have different numbers of parameters or parameters of different types. Java compiler differentiate these methods on the number and type of parameters, allowing you to have multiple methods with same name but different behaviors.

Eg:-
```
Class MethodOverloading{
    private static void display (int a){
        System.out.println ("Arguments:" +a);
    }
    private static void display (int a, int b){
        System.out.println ("Arguments" +a+
                            " and " + b);
    }
    public static void main(String[] args)
    {
        display (1);
        display (1, 4);
    }
}
```

Date .....
Page .....

Q2) What are the rules for method overloading resolution in Java? How does Java determine which overloaded method to call?

Ans) The rules for method overloading:

1) Number of parameters:- Java selects the overloaded method with the same number of parameters as the arguments provided in the method call. If multiple overloaded methods have the same number of parameters future rules are applied.

2) Type of parameters:- If multiple methods have the same number of parameters, Java chooses the one with parameters that much the data type of the arguments most closely

3) Type promotion:- Java promotes arguments to larger data type if an exact match is not found.

4) Autoboxing and varargs:- Java prefers autoboxing and varags over widening conversions.

5) Inheritance:- If a method is overloaded in a subclass, Java chooses the

Subclass method over the superclass method if applicable.

6] Varags vs non-varargs :- If there's chance between a varg varargs and non-varargs method Java prefers the non-varargs method.

Java aims to select the most specific overloaded method based on the provided argument while adhering to these rules. If no unique method can be determined, a compilation error occurs due to ambiguity.

Q] what does the Static Keyword mean in Java? Explain the difference between static and non-static method.

Ans] In Java the 'static' Keyword is used to define a member of a class that belong to the class itself rather than to instances of the class.

1] Static Methods:-

- Belong to the class itself, not to instances of the class.
- Can be called directly using the class name, without creating an instance of the class.

Date .....
Page .....

- Cannot access instance variable directly as they belong to instances of the class
- Commonly used for utility methods, constants, or methods that do not rely on instancespecific data.

2. Non-Static Methods:-
- Belongs to individual instances of the class.
- Can access both static and instance variables and methods
- Must be called on an instances of the class
- Represents behaviour or actions specific to individual instances of the class.

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

Ans] Yes, Satic methods can be overloaded but not overridden in Java.
- Overloading:- Static method can be overloaded refers to defining multiple method in the same class with the same name but with different parameters. The

Compiler determines which method to call based on the parameters passed.

Overriding :- Static methods cannot be overridden in Java. When a subclass defines a static method ~~in Java~~ with the same signature as a static method in the superclass, it hides the superclass method rather than overriding it. The method called is determined by the reference type at compile time rather than the object ~~by~~ type at runtime.

Static variables are shared across multiple instances of a class ~~be~~ itself rather than to individual instances. All instances of the class share the same copy of static variables. Changes made to static variables through an instance will be reflected in all other instances and in the class itself.

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

Ans) In the context of memory management, the 'static' keyword in Java indicates that a variable or method is associated with the class rather than with individual instances of the class.

- Static variables: Memory of static variables is alloated once when the class is loaded into memory and exist for the entire lifetime of the program. These variables are stored in the method area of the JVM (Java virtual Machine) memory.
- Static ~~memory~~ methods :- Similarly, static methods are loaded into memory when the class is loaded and can be called without creating an instance of the class. They also exist for the entire lifetime of the program.

- By associating variables and methods with the class rather than with individual instances, the 'static' keyword helps in efficient memory management by reducing ~~method~~ memory consumption and avoiding necessary duplication of

data across multiple instances of class.

Q) What is the significance of the final keyword in Java?

Ans) In Java the 'Final' keyword is used to define constants, prevent method overriding, and restrict subclassing.

- Constants :- When applied to a variable, it indicates that its value cannot be changed after ~~init~~ initialization.
- Methods :- When applied to a method, it prevent subclasses from overriding that method.
- Classes :- when applied to a classes, it prevents the class from being subclassed.

In short the final keyword in Java ensures immutability prevents method overiding, and restricts subclassing.

Q) Can a final method be overidden in a subclass? How does the final keywords affect variables, methods, and classes in Java?

Ans) No, a final method cannot be overridden in a subclass in Java.

- Variables:- when applied to variables, 'final' makes them constants, meaning their values cannot be changed after initialization.
- Method:- when applied to method 'final' prevents subclasses from overriding that method
- Classes: when applied to classes 'final' prevents the class from being subclassed.

In short 'final' keyword in Java ensure immutability for variables, prevents method overriding and restricts subclassing.

Q) What does the keyword represent in Java? How is the keyword used in constructors and methods?

Ans) In Java 'this' keyword is a reference to the current object within a class. It can be used in the following ways:

- Constructor: In a constructor 'this' is used to differentiate between instance variable and parameters with the same name. It is used to

assign values to instance variable or to call another constructor from within the same class.

- Methods: In method, 'this' is used to access instance variables and methods of the current object. It can be used to invoke methods or access instance variables, explicitly, especially when there's naming conflict with local variables.

In short the 'this' keyword in java refers to the current object and is used to access its members, resolve naming conflicts, and call constructors within the same class.

Q) What are narrowing and widening conversions in Java?

Ans) In Java narrowing and widening conversion refers to the conversion of data types between wider and narrower ranges

- Widening Conversion:- Involves converting a data type to a larger data type to accommodate a wider range of values without the risk of losing information. For example:- converting an 'int' to a 'double'.

Date
Page

- Narrowing Conversion:- Involves converting a data type to a smaller data type, potentially resulting in the loss of information or precision. for example, converting a 'double' to an 'int'.
In short, widening conversion expand data types to accommodate larger values, while narrowing conversions shrink data types, possibly resulting in data loss.

Q) Provide example of narrowing and widening conversion between primitive data type in short

Ans) Widening Conersion:-

```
int intvalue = 10;
double doublevalue = intvalue;
//Widening Conversion from int to double
System.out.println(doublevalue); //output:10.0
```

Narrowing Conversion from double to int.

```
double doublevalue = 10.5;
int intvalue = (int) doublevalue;
system.out.println(intvalue); //o/p: 10
```

In the narrowing conversion example, the fraction part of the 'double' value is truncate when it's converted to an 'int', resulting in data loss.

Q) How does the java handle potential loss of precision during narrowing conversion?

Ans) Java handle potential loss of precision during narrowing conversions by requiring explicit casting. This informs the compiler that the programmer is aware of the potential loss and intends to proceed with the conversion, allowing the conversion to occur but truncating any fractional part without performing rounding.

Q) Explain the concept of automatic widening Conversion in Java.

Ans) Automatic ~~convert~~ widening conversion in Java refers to the implicit conversion of data types from a narrower type to a wider type without requiring explicite casting. Java automatically widens data types to accommodate larger values safely, ensuring no loss

of information.
For example, when assigning a smaller data type to a larger one, such as assigning an 'int' to a 'double' Java performs automatic widening conversion:

eg:-
```
int invalue = 10;
double doubleValue = intValue // Automatic
    // widening conversion from int to double
```

In this case the 'int value is ~~out~~ automatically widened to a 'double' without the need for explicit casting. Automatic widening conversions are safe and do not result in loss of information.

Q) What are the implications of narrowing and widening conversions on type compatibility and data loss?

Date
Page

Ans) Widening Conversions: Widening conversion expands data types to accomodate longer values without loss of information. They ensure type compatibility and do not result in data loss.

- Narrowing Conversions:- Narrowing conversions shrink data type, potentially resulting in data loss. They made violate type compatibility and require explicit casting to proceed making it crucial for the programmer to handle potential loss of information carefully.