Name: Manasi Kailas Borade.

PRN:240340320060

Assignment No- 6

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

Method overloading in Java refers to the ability to define multiple methods in a class with the same name but with different parameter lists. This allows a single method name to represent different behaviors based on the parameters passed to it. The parameters can differ in the number, type, or order.

```
class Vehicle{

   public static void vehicleDetails(String vname ,String vmodel){
          System.out.printf("%s   %s  \n",vname,vmodel);
   }
    public static void vehicleDetails(String vname ,float price){
          System.out.printf("%s   %.2f \n",vname, price);
   }
    public static void vehicleDetails(String color ){
          System.out.printf("%s   ",color);
   }
   /*public static void vehicleDetails(String vname,String color ){
          System.out.printf("%s  %s",vname,color);
          (string,string) already defined in class vehicle
   }
   */


   public static void main(String args[]){
     Vehicle.vehicleDetails("Car","Maruti-TATA-25Z");
     Vehicle.vehicleDetails("Bus",2000.0f);
     Vehicle.vehicleDetails("BLACK");
      Vehicle.vehicleDetails("RED","Bike");
   }

//static methods===> by using class name
}
```

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

a. Number of Parameters: The compiler checks the number of parameters provided in the method call. It selects the overloaded method with a matching number of parameters.

b. Parameter Type: If the number of parameters matches, the compiler checks the types of parameters in the method call. It selects the overloaded method with parameter types that exactly match the types of the arguments in the method call. If an exact match is not found, the compiler looks for the closest match based on the method invocation context. The Java compiler will attempt to convert the arguments to the parameter types if possible, choosing the most specific applicable version.

c. of Parameters: If there are multiple overloaded methods with the same number and types of parameters but in different orders, the compiler selects the method that matches the order of arguments in the method call.

3) What does the static keyword mean in Java? Explain the difference between static and non-static methods.

In Java, the static keyword is used to declare members that belong to the class rather than to instancesof the class. When a member is declared as static, it means there is exactly one copy of that member shared among all instances of the class.

a)static method->
1.Belong to the Class
2.Cannot Use "this" Keyword

b)non static method->
1.Belong to Instances:
2.Use "this" Keyword

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

static methods can be overloaded but not overridden.

When a variable is declared as static, memory for that variable is allocated only once, regardless of how many instances (objects) of the class are created. This memory allocation occurs when the class is loaded into memory by the Java Virtual Machine (JVM). Static variables are stored in a separate area of memory known as the "Method Area".Since there is only one copy of a static variable shared among all instances of the class,

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

Static Keyword-> When a variable is declared as static, memory for that variable is allocated only once, regardless of how many instances (objects) of the class are created. This memory allocation occurs when the class is loaded into memory by the Java Virtual Machine (JVM). Static variables are stored in a separate area of memory known as the "Method Area".Since there is only one copy of a static variable shared among all instances of the class,

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

Final-> When applied to a variable, the final keyword indicates that the variable’s value cannot be changed after it has been initialized. Once assigned a value, a final variable becomes a constant and cannot be reassigned.Final keyword is used to denote that something is unchangeable or cannot be overridden,

7)Can a final method be overridden in a subclass?How does the final keyword affect variables, methods, and classes in Java?

No, a final method cannot be overridden in a subclass. If a method is declared final in a superclass, it means it cannot be overridden by any subclass.

Final Variables:

When applied to a variable, it means that the variable's value cannot be changed once initialized.
If the variable is a primitive type, its value cannot be altered.If the variable is a reference to an object, it cannot be reassigned to point to a different object. However, the state of the object itself can be modified.

Final Methods:

When applied to a method, it means that the method cannot be overridden by subclasses.
This is useful when you want to enforce a specific behavior in a class hierarchy and prevent subclasses from changing it.

Final Classes:

When applied to a class, it means that the class cannot be subclassed.
This is useful when you want to prevent inheritance and ensure that the class remains as-is.

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

In Java, the this keyword is a reference to the current instance of the class in which it appears. It can be used to access instance variables, invoke methods, and invoke constructors within the same class. The primary purpose of this is to disambiguate between instance variables/methods and local variables/method parameters when they have the same name.

Invoking Other Constructors: In a constructor, this() can be used to call another constructor in the same class. This is known as constructor chaining. It must be the first statement in the constructor.

9) What are Widening and Narrowing conversions in Java?

Widening-> (Implicit Conversion) Widening conversion occurs when you convert a value from a data type with a smaller range to a data type with a larger range.
In widening conversion, there is no risk of losing information because the target data type can represent all possible values of the source data type.

Narrowing->(Explicit Conversion)Narrowing conversion occurs when you convert a value from a data type with a larger range to a data type with a smaller range.
In narrowing conversion, there is a risk of losing information because the target data type may not be able to represent all possible values of the source data type.

10) Provide examples of widening and narrowing conversions between primitive data types.

Widening->

```
class Widening
{
    public static void main(String args[]){
    int num=5;
    double num1=num;
    System.out.println(num1);
```

```
    }
}
```

Narrowig->

```
class Narrowing
{
        public static void main(String args[]){
        double num=25.00;
        int num1=(int)num;
        System.out.println(num1);
    }
}
```

11) How does Java handle potential loss of precision during narrowing conversions?

Java handles potential loss of precision during narrowing conversions by truncating the higher-order bits of the value being converted. When you convert a value from a data type with a larger range to a data type with a smaller range, Java discards the extra bits that cannot be represented in the smaller data type. This process results in potential loss of precision, as some information from the original value may be lost.

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

Widening-> (Implicit Conversion) Widening conversion occurs when you convert a value from a data type with a smaller range to a data type with a larger range.
In widening conversion, there is no risk of losing information because the target data type can represent all possible values of the source data type.

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

Type Compatibility:

Widening Conversion: Widening conversions involve converting a value from a smaller data type to a larger data type. Since the target data type can accommodate a wider range of values, there's no loss of precision, and such conversions are always allowed without explicit casting. For example, converting an int to a double.

Narrowing Conversion: Narrowing conversions involve converting a value from a larger data type to a smaller data type. These conversions can lead to loss of precision, as some information may be discarded to fit the value into the smaller data type. Narrowing conversions require explicit casting, and they should be performed with caution to avoid unintended loss of information.

Data Loss:

Widening Conversions: Widening conversions typically do not result in data loss because the target data type can represent all possible values of the source data type. For example, converting an int to a double preserves the full precision of the int.

Narrowing Conversions: Narrowing conversions can lead to data loss, especially when converting from a floating-point type to an integer type, or when the source value exceeds the range of the target data type.

For example, converting a double with a fractional part to an int results in the loss of the fractional part.