Assignment No. 6

Q1. what is the method overloading in Java & explain with an example?

Sol. Method overloading in Java is the process that can create multiple methods of the same name in the same class, and all the methods work in different ways.

eg.

```
~~class sha~~ public class MethodOverloadingExample{
    // Method to add two integers
    public static int add (int a, int b) {
        return a+b;
    }
    // Method to add three integers
    public static int add (int a, int b, int c){
        return a+b+c;
    }
    // Main method
    public static void main(String [] args){
        // Method calls demonstrating method
                                    overloading
        System.out.println("Sum of 5 and 7:"
                              + add (5,7));
        System.out.println("Sum of 2,3, and 4:"
                          +add (2,3,4));
    }
}
```

Page no.: ____/

Date: __/__/____/

Q2. What are the rules for method overloading resolution in Java? How does Java determines which overloading method to call?

Sol. - In overloading, methods must have different parameters

- The class should be same
- The method should be same.
- The parameter type or parameter number should be different.

Java determining method overloading based on the no. and types of parameter in the method signature.

Q3. What does static keyword mean in Java? Explain the difference b/w static & non static methods.

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

Static method is a class method and belongs to the class itself. This means you do not need an instance in order to use a static method and belongs to each object that is generated from the class.

Q4. Can static methods be overloaded and overridden in Java? How are static variables shared multiple instances of a class?

Sol. Static methods in Java cannot be overriden.

This is because static methods are not associated with the instance of a class, but with the class itself. Therefore, when a subclass inherits a static method from its parent class, it cannot modify the behaviour of the satic method in any way.
If the same class is loaded in different class loaders, then each copy of the class will have its own statics.

Q5. What is the role of static Keyword in the context of memory management?

Sol. 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 Keywords in variables, methods, blocks, and nested classes. The static keyword belongs to the class than an instances, reducing the amount of memory required.

Q6. What is the significance of final Keyword in Java?

Sol. final is a Keyword in Java.
① final variable - If we make any variable as final we cannot change the value of final variable (it will be constant)
② final method - If we make method as final, we cannot override it.
③ Final class - If we make any class a final we cannot extend it.

Page no.: ____/
Date: ___/___/_____

Q7. Can a final method be overriden in a subclass? How does the final keyword affect variable, methods, & classes in Java?

Sol. No the methods that are declared as final cannot be overridden or hidden.
If variables made final then we cannot change its values. If method made final then we cannot override it. If classes made final then we cannot extend it.

Q8. What does 'this' keyword represent in Java? How is this keyword used in constructors and methods?

Sol. 'this' represents the instance of the class where it's used. It's commonly used to access or modify the fields of the current object, esp. when field names are the same as local variables names.
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 b/w class attributes and parameters with the same name.

Q9. What are narrowing & widening conversions in Java?

1. In Java there are two types of casting:
① Widening - Converting a smaller data type to a larger data type.
eg. byte → short → char → int → long.
② Narrowing - Converting a larger data type to a smaller data type.

eg: float → long → int → char → short → byte.

Q10. Provide examples of narrowing & widening conversions b/w premitive data types?

Sol. Widening –

```
        public class Main {
                public static void main (string[]args){
          int a = 5;
          float b=3.5f
          float sum = a+b;
        System.out.print ln ("values of a= " +a);
        System.out.print ln ("values of b= "+ b);
        System .out.print ln (" sum = "+ sum);
      }
   }
```

Narrowing –

```
        public class Main {
                 public static void main (String []args){
          float a= 5.7f;
          int b= (int)a;
        System.out.println ("Values of b= "+b);
          }
      }
```

Page no.:____/

Date:__/__/___/

Q11: How does Java handle potential loss of precision during narrowing conversions?

Sol. In the case of double to float, you can have a constant value which is in the right range but still lose precision. In your specific case of 10.0, the value can be represented exactly in both float and doubly.

```
eg. float f = (float) 10.1;
    double d = 10.1;
    System.out.println (f = -d);
```

Q12. Explain the concept of automatic widening conversion in Java.

Sol. Widening conversion take place when two data types are automatically converted. This happens when the two data types are compatible. When we assign a value of a smaller data type to a bigger data type.

byte → short → int → long → float → double.

Q13. What are the implications of narrowing and widening conversions on type compatibility and data ~~ty~~ loss?

Sol. Widening conversions preserve the source ~~of~~ value but can change its representation. This occurs if you convert from an integral type to Decimal, or from char to string. A narrowing conversion changes a value to a data type that might not be able to hold some of the possible values.