Q.1What is difference betweem method prototype And method signature?

Ans:

The method signature defines how the method is uniquely identified in a class.

1.Method Signature:

Method name

Parameter list (type and order)

Important: It does not include return type, throws clause, or access modifiers (public, private, etc.)

Example:

```
void add (int a, int b) {
}  // Signature: add (int, int)

int add (int x, int y) {
}   // Signature is still add(int, int)
```

2. Method Prototype

A method prototype is a declaration of the method without its body. It tells the compiler what the method will look like, including:

a) Access modifiers (public, private, etc.)

b) Return type

c)Method name

4)Parameter types

5)throws clause (optional)

Example:

public int add (int a, int b);  // Method prototype (no body)

Method prototype is common in interfaces and abstract classes in Java.

It represents the complete declaration, while the method signature is just the "name + parameters" part.

Q.2 Main Method with Final Keyword

Ans:

```
class DemoExampleForFinalKeyword{

final public static void main(String [] args){    //Declaring Main Method As a final
System.out.println("Printing main method by using  final keyword");

}
}
```

O/P:

```
C:\JAVAProgams>javac DemoExampleForFinalKeyword.java

C:\JAVAProgams>java DemoExampleForFinalKeyword
Printing main method by using  final keyword

C:\JAVAProgams>
```

we can use final before the main method

Q.3 Wap for overloading main method?

Ans:

```
class DemoExample{
 public static void main(String [] args){
System.out.println("Main Method ");

}
public static void main(int a){
        System.out.println("Overloaded main Method");

}
public static void main(char c,String str){
        System.out.println("Overloaded main Method");

}
}
```

O/p:

```
/*C:\JAVAProgams>javac DemoExample.java

C:\JAVAProgams>java DemoExample
Main Method

C:\JAVAProgams>*/
```

It will print message declared inside the first main method not others because ,Jvm has a fixed entry point: public static void main(String[] args)

Overloaded methods are just normal static methods, not recognized automatically as entry points.

Q.4.Wrie a clean code which will show the correct NamingConvention

```
// Class names: Should start with uppercase and follow CamelCase
public class NamingConventionExample {

   // Constants: Should be uppercase with underscores
   public static final double FI = 878.787;

   // Instance variable: Should start with lowercase and follow camelCase
   int studentAge;

   // Static variable
   static String schoolName = "Xyz High School";

   // Constructor
   public NamingConventionExample(int age) {
      this.studentAge = age;
   }

   // Method names: Should start with lowercase and follow camelCase
   public void displayStudentInfo() {
      System.out.println("Student Age: " + studentAge);
      System.out.println("School Name: " + schoolName);
   }

   // Main method
   public static void main (String [] args) {
      // Local variables: camelCase
      int studentRollNo = 101;
```

```
        // Object creation
        NamingConventionExample student = new NamingConventionExample(16);

        // Method call
        student.displayStudentInfo();

        // Printing local variable
        System.out.println("Student Roll Number: " + studentRollNo);

        // Constant usage
        System.out.println("Value of PI: " + FI);
    }
}
```

```
C:\JAVAProgams>javac NamingConventionExample.java
C:\JAVAProgams>java NamingConventionExample
```

O/P:

```
Student Age: 16
School Name: Xyz High School
Student Roll Number: 101
Value of PI: 878.787
```