List and explain the various contexts of using final keyword in Java with detailed programming explanation ?

FINAL KEYWORD :

In Java, the final keyword can be used in different contexts to indicate that an entity (variable, method, or class) cannot be changed or overridden.

Final Variables:

Definition: Final variables are variables whose values cannot be changed once assigned. They act as constants in Java programs.

Syntax:

```
final dataType variableName = value;
```

Example:

```
public class FinalVariableExample {
    final int MAX_VALUE = 100;

    public static void main(String[] args) {
        FinalVariableExample obj = new FinalVariableExample();
        System.out.println("Maximum value: " + obj.MAX_VALUE);
    }
}
```

EXPLANATION:

In this example, **MAX_VALUE** is declared as a final variable and initialized with the value **100**. Its value cannot be changed after initialization.

2. Final Methods:

Definition: Final methods are methods that cannot be overridden by subclasses. They provide consistency in behavior across the class hierarchy.

Syntax:

```
final returnType methodName() {
   // Method body
}
```

EXAMPLE:

```
class Bike{
 final void run(){System.out.println("running");}
}

class Honda extends Bike{
  void run()
 {
   System.out.println("running safely with 100kmph");
}

 public static void main(String args[])
{
  Honda honda= new Honda();
  honda.run();
  }
}
```

```
Output:Compile Time Error
```

Explanation: The run**()** method in the **Parent** class is marked as final, preventing any subclass from overriding it.

3. Final Classes:

Definition: Final classes are classes that cannot be subclassed. They are used to prevent further specialization of a class.

Syntax:

final class ClassName {

// Class members and methods }

EXAMPLE:

```
final class Bike{}

class Honda1 extends Bike{
 void run(){System.out.println("running safely with 100kmph");}

 public static void main(String args[]){
 Honda1 honda= new Honda1();
 honda.run();
      }
   }
```

```
Output:Compile Time Error
```

4. Final Parameters:

Definition: Final parameters are method parameters whose values cannot be changed within the method. They ensure that the parameter value remains constant throughout the method execution.

Syntax:

```
void methodName(final dataType parameterName) {
   // Method body
}
```

EXAMPLE:

```
class Bike11{
 int cube(final int n){
  n=n+2;//can't be changed as n is final
  n*n*n;
 }
 public static void main(String args[]){
   Bike11 b=new Bike11();
   b.cube(5);
 }
}
```

```
Output: Compile Time Error
```

5. Final Fields in Classes:

Definition: Final fields in classes are fields that must be initialized exactly once, either when declared or in the constructor, and cannot be modified afterward.

Syntax:

```
final dataType fieldName;
```

EXAMPLE:

```
public class FinalFieldsExample {
    // Final field initialized when declared
    final int MAX_VALUE = 100;

    // Method to display the value of the final field
    void displayMaxValue() {
        System.out.println("Maximum Value: " + MAX_VALUE);
    }

    public static void main(String[] args) {
        // Creating an instance of the FinalFieldsExample class
        FinalFieldsExample obj = new FinalFieldsExample();

        // Displaying the value of the final field
        obj.displayMaxValue();
    }
}
```

Explanation:

- In this example, **`MAX_VALUE`** is declared as a final field within the **`FinalFieldsExample`** class and initialized with the value **`100`**.
- Once initialized, the value of the final field cannot be changed for the lifetime of the object.
- The **`displayMaxValue()`** method is used to display the value of the final field.
- In the **`main()`** method, an instance of the **`FinalFieldsExample`** class is created, and the value of the final field is displayed.

6. Final Arrays:

Definition: Final arrays are arrays whose reference cannot be changed once initialized, but the elements within the array can still be modified.

Syntax:

final dataType[] arrayName = { /* Array elements */ };

EXAMPLE:

```
public class FinalArrayExample {
   final int[] numbers = {1, 2, 3, 4, 5};
   public static void main(String[] args) {
      FinalArrayExample obj = new FinalArrayExample();
      obj.numbers[0] = 10; // Modifying array element is allowed
      System.out.println("First element: " + obj.numbers[0]);
   }
}
```

Explanation: In this example, the **`numbers`** array is declared as final, meaning its reference cannot be changed. However, individual elements within the array can still be modified.

7. Blank Final Variables:

Definition: Blank final variables are final variables that are not initialized when declared but must be initialized exactly once either in a constructor or an instance initialization block before they are used.

Syntax: final dataType variableName;

EXAMPLE:

```
class Student{
int id;
```

```
String name;
final String PAN_CARD_NUMBER;
...
}
```

Can we initialize blank final variable?

Yes, but only in constructor. For example:

```
class Bike10{
  final int speedlimit;//blank final variable

  Bike10(){
  speedlimit=70;
  System.out.println(speedlimit);
  }

  public static void main(String args[]){
    new Bike10();
  }
}
```

```
Output: 70
```

8. Static Final Variables:

Definition: Static final variables are constants that belong to the class itself rather than to instances of the class. Once initialized, their value remains constant for the entire duration of the program.

Syntax:

```
static final dataType VARIABLE_NAME = value;
```

EXAMPLE:

```
public class StaticFinalExample {
   // Declaring a static final variable for the value of PI
   static final double PI = 3.14;


   // Method to calculate the area of a circle using the static final variable
   static double calculateCircleArea(double radius) {
      return PI * radius * radius;
   }
public static void main(String[] args) {
      double radius = 5.0;
      // Calculating and displaying the area of the circle
      double area = calculateCircleArea(radius);
      System.out.println("Area of the circle with radius " + radius + ": " + area);
      // Attempting to modify the value of the static final variable (which is not allowed)
      // PI = 3.14159; // Error: cannot assign a value to final variable PI
   }
}
```

Explanation:

- In this example, **`PI`** is declared as a static final variable, representing the value of pi (π).
- The **`calculateCircleArea()`** method utilizes the static final variable **`PI`** to compute the area of a circle.

- In the **`main()`** method, we calculate and display the area of a circle with a given radius.
- Additionally, an attempt to modify the value of the static final variable **`PI`** is made, which results in a compilation error, demonstrating that static final variables cannot be reassigned once initialized.