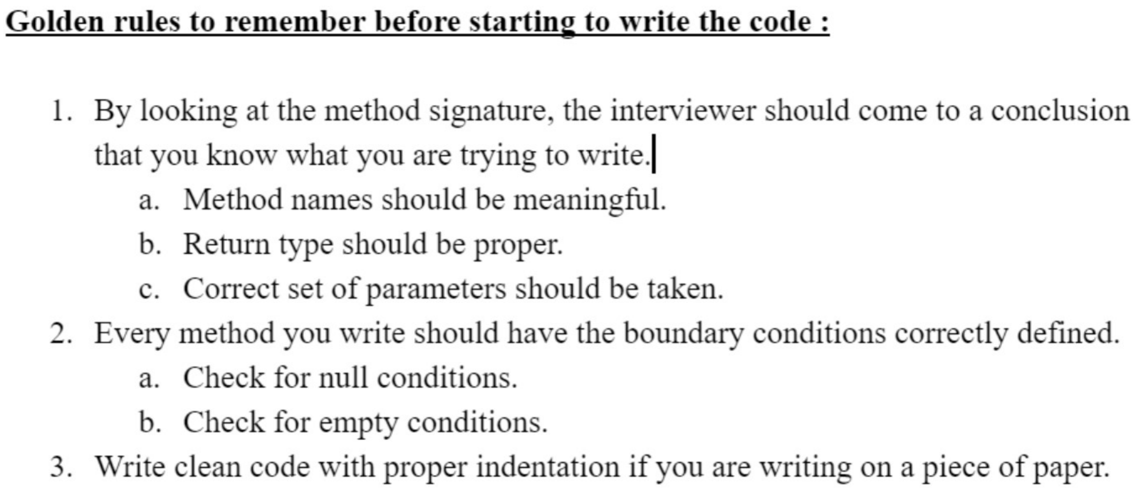


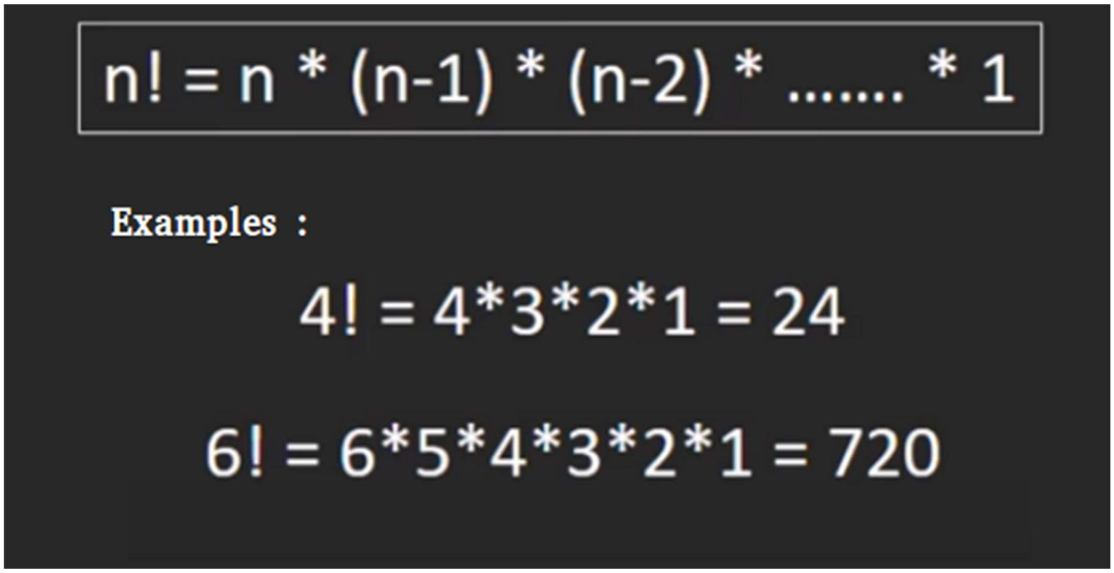
Golden rules to remember before starting to write the code :

1. By looking at the method signature, the interviewer should come to a conclusion that you know what you are trying to write.
    a. Method names should be meaningful.
    b. Return type should be proper.
    c. Correct set of parameters should be taken.
2. Every method you write should have the boundary conditions correctly defined.
    a. Check for null conditions.
    b. Check for empty conditions.
3. Write clean code with proper indentation if you are writing on a piece of paper.

Example code following above rules:

```java
public class Employee {
    int empId;
    String empName;

    public Employee(int empId, String empName) {
        super();
        this.empId = empId;
        this.empName = empName;
    }

    public int getEmpId() {
        return empId;
    }

    public void setEmpId(int empId) {
        this.empId = empId;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + empId;
        result = prime * result + ((empName == null) ? 0 : empName.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Employee other = (Employee) obj;
        if (empId != other.empId)
            return false;
        if (empName == null) {
            if (other.empName != null)
                return false;
        } else if (!empName.equals(other.empName))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "Employee [empId=" + empId + ", empName=" + empName + "]";
    }

}

public class EmployeeTest {
    public static Employee changeName(Employee emp, String name) {
        // Base condition 1
        if (emp == null) {
            return null;
        }
        // Base condition 2
        if (name == null) {
            return emp;
        }
        // Base condition 3
        if (name.isEmpty()) {
            emp.setEmpName("Empty");
            return emp;
        }
        // Actual logic.
        emp.setEmpName(name);
        return emp;
    }

    public static void main(String[] args) {
        Employee e1 = new Employee(1, "");
        Employee changedEmployee = changeName(e1, "");
        System.out.println(changedEmployee);
    }

}
```

Factorial Implementation: -



n! = n * (n-1) * (n-2) * .... * 1

Examples :

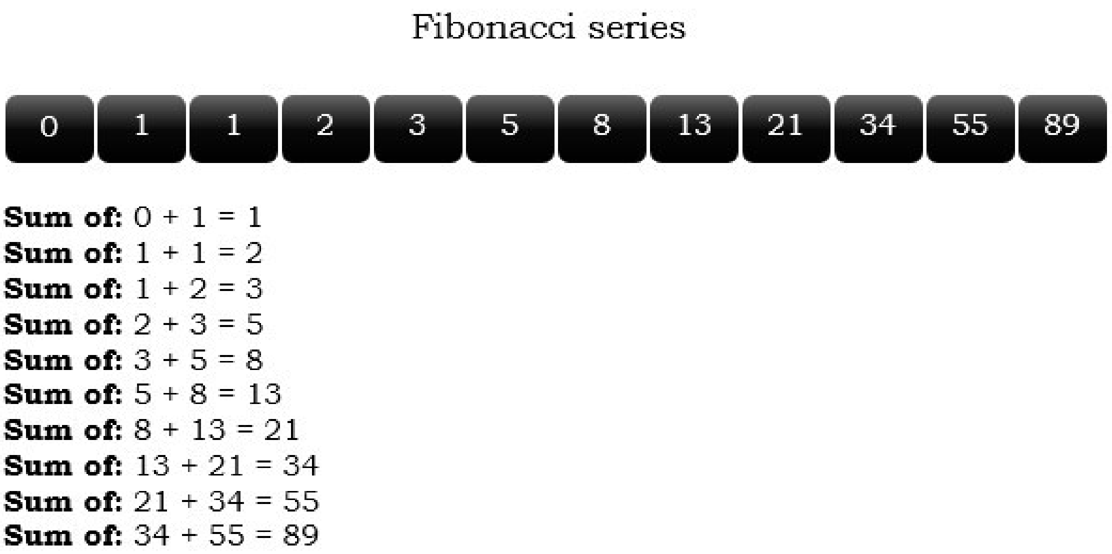
4! = 4*3*2*1 = 24

6! = 6*5*4*3*2*1 = 720

```java
public class FactorialIterative {
	public static void main(String[] args) {
		System.out.println(fact(6));
	}

	static int fact(int n) {
		if (n == 0 || n == 1) {
			return 1;
		}
		int result = 1;
		for (int i = n; i >= 2; i--) {
			result *= i;
		}
		return result;
	}
}
```

Time Complexity: O(n) : Space Complexity : O(1)

<u>Fibonacci Series implementation: -</u>



Fibonacci series

0 1 1 2 3 5 8 13 21 34 55 89

Sum of: 0 + 1 = 1
Sum of: 1 + 1 = 2
Sum of: 1 + 2 = 3
Sum of: 2 + 3 = 5
Sum of: 3 + 5 = 8
Sum of: 5 + 8 = 13
Sum of: 8 + 13 = 21
Sum of: 13 + 21 = 34
Sum of: 21 + 34 = 55
Sum of: 34 + 55 = 89

```
public class FibonacciSeriesIterative {

	public static int fib(int n) {
		int a = 0;
		int b = 1;
		int c = 1;
		System.out.print(a + "," + b);
		for (int i = 1; i <= n; i++) {// Iteration starts form 1 and not 0.
			a = b;
			b = c;
			c = a + b;
			System.out.print("," + c);
		}
		return c;
	}

	public static void main(String[] args) {
		fib(7);
	}
}
```

Time Complexity: O(n) : Space Complexity : O(1)