**JAVA THREADING ASSIGNMENT**

**3rd June 2025**

**Abhik Chakraborty (**[**Abhik.Chakraborty@bounteous.com**](mailto:Abhik.Chakraborty@bounteous.com)**)**

1. **Print Odd and Even Numbers Using Two Threads**

You are tasked with writing a Java program that prints odd and even numbers in sequence using two separate threads. One thread should print odd numbers, and the other thread should print even numbers. The threads should coordinate to ensure the numbers are printed in the correct order.

**a. Requirements**

**i. Two Threads**: One thread prints odd numbers, and the other thread prints numbers.

**ii. Synchronization:** The threads must coordinate to print the numbers in sequence without any overlap or missing numbers.

**iii. Range:** The program should print numbers from 1 to a specified maximum value.

**Implementation**

**· OddEvenPrinter Class:** Manages the printing of odd and even numbers using two threads.

**· OddThread Class:** Represents the thread that prints odd numbers.

**· EvenThread Class:** Represents the thread that prints even numbers.

Solution:

Print Odd and Even Numbers Using Two Threads

Print numbers in order from 1 to N using two threads – one for odd, one for even.

Parent Class -> OddEvenPrinter

Child Classes which extends form Thread Class

->OddThread

->EvenThread

OddEvenPrinter.java



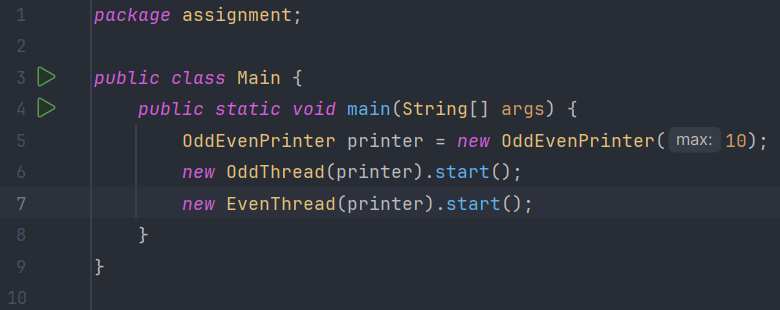
OddThread.java

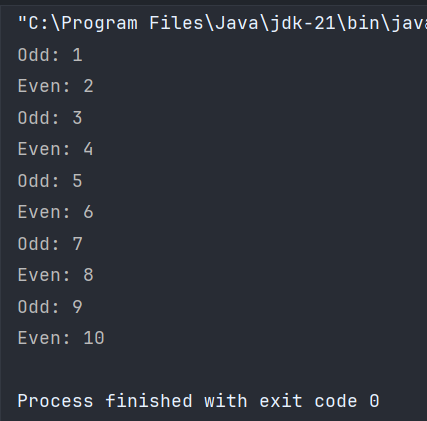
****

EvenThread.java

****

Main Class



OUTPUT:  


1. **Bridge Crossing with Shared Token**

There are two cities, City A and City B, connected by a bridge. Only one person can cross the bridge at a time. To cross the bridge, a person must take a token from one end and deposit it at the other end. There is only one token available, and it must be shared by all residents of both cities. Initially, the token is in City B. Residents of City B must use the token to travel to City A first, and only then can residents of City A use the token to travel to City B.

**Requirements**

**1. Bridge:** A shared resource that only one person can use at a time.

**2. Token:** A token with one permit representing the token that controls access to the bridge.

**3. Direction Control:** A mechanism to ensure that the token is used by residents of City B to travel to City A first, and then by residents of City A to travel to City B.

**4. Implement Waiting Queue:** Implement queue for both cities so that people will get the turn to cross the city.

**Implementation**

**1. Bridge Class:** Manages the token and the direction control.

**2. Person Class:** Represents a person who wants to cross the bridge.

**3. BridgeManagement Class:** Creates instances of Person and starts their threads.

Solution:

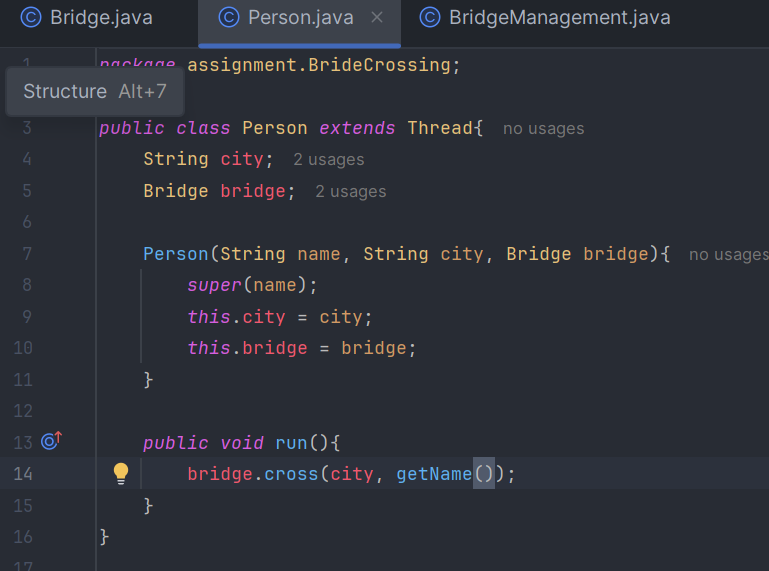
**Bridge Crossing with Shared Token**

People from two cities share a bridge with a token. Only one crosses at a time.

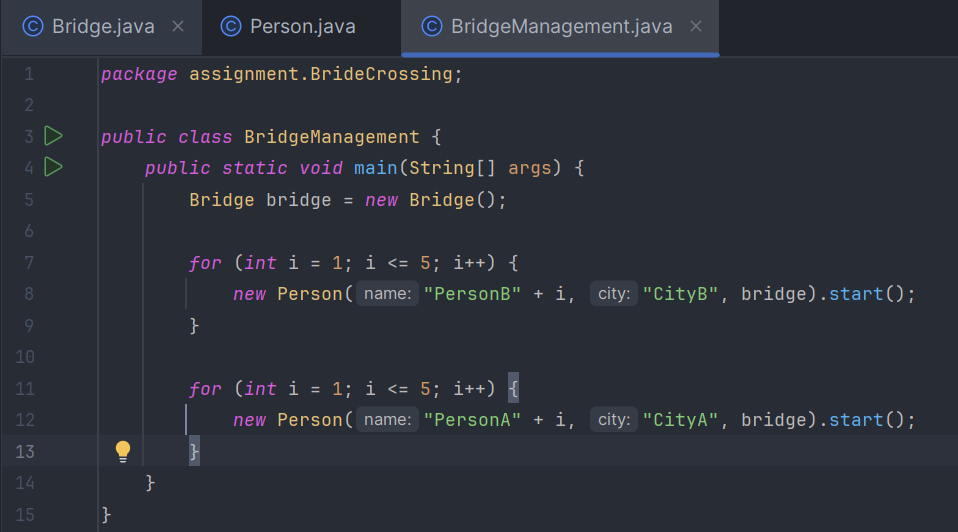
Bridge ->parent class

Person -> extends from Thread

Person.java



BridgeManagement.java



OUTPUT:

