

Java Assignment-5 Streams and Lambda

1. Write the following methods that return a lambda expression performing a specified action:

PerformOperation isOdd(): The lambda expression must return if a number is odd or if it is even.

PerformOperation isPrime(): The lambda expression must return if a number is prime or if it is composite.

PerformOperation isPalindrome(): The lambda expression must return if a number is a palindrome or if it is not.

Input Format

Input is handled for you by the locked stub code in your editor.

Sample Input

The first line contains an integer, (the number of test cases).

The subsequent lines each describe a test case in the form of space-separated integers:

The first integer specifies the condition to check for (for Odd/Even, for Prime, or for Palindrome).

The second integer denotes the number to be checked.

```
5
1 4
2 5
3 898
1 3
2 12
```

Sample Output

```
EVEN
PRIME
```

PALINDROME
ODD
COMPOSITE
Language

Program:

```
import java.util.Scanner;

public class NumberChecker {
    public static PerformOperation isOdd() {
        return n -> n % 2 != 0;
    }

    public static PerformOperation isPrime() {
        return n -> {
            if (n <= 1) return false;
            if (n <= 3) return true;
            if (n % 2 == 0 || n % 3 == 0) return false;
            for (int i = 5; i * i <= n; i += 6) {
                if (n % i == 0 || n % (i + 2) == 0) return false;
            }
            return true;
        };
    }

    public static PerformOperation isPalindrome() {
        return n -> {
            String str = String.valueOf(n);
            String reversed = new StringBuilder(str).reverse().toString();
            return str.equals(reversed);
        };
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int testCases = scanner.nextInt();

        while (testCases-- > 0) {
            int condition = scanner.nextInt();
```

```

int number = scanner.nextInt();
PerformOperation operation = null;

switch (condition) {
    case 1:
        operation = isOdd();
        break;
    case 2:
        operation = isPrime();
        break;
    case 3:
        operation = isPalindrome();
        break;
}

String result = operation.check(number) ? "YES" : "NO";
System.out.print(result + " ");
}
scanner.close();
}
}

```

```

@FunctionalInterface
interface PerformOperation {
    boolean check(int a);
}

```

Sample Input:

```
5
1 4
2 5
3 898
1 3
2 12
```

Sample Output:

```
EVEN PRIME PALINDROME ODD COMPOSITE
```

2. Write a Java program for implementing Runnable using Lambda expression

```
public class RunnableLambdaExample {
    public static void main(String[] args) {
        // Using Lambda expression to create a Runnable
        Runnable myRunnable = () -> {
            for (int i = 0; i < 5; i++) {
                System.out.println("Thread is running: " + i);
            }
        };

        // Create and start a new Thread using the Runnable
        Thread myThread = new Thread(myRunnable);
        myThread.start();
    }
}
```

RESULT

```
Thread is running: 0  
Thread is running: 1  
Thread is running: 2  
Thread is running: 3  
Thread is running: 4
```

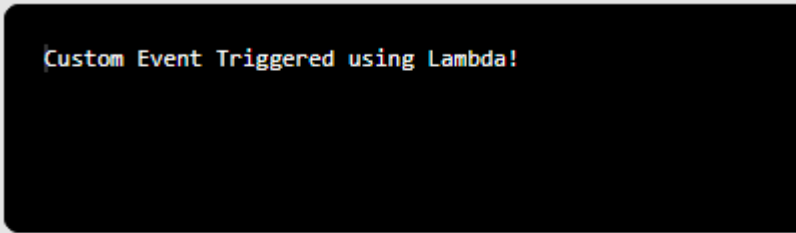
3. Write a Java program for event handling using Java 8 Lambda expressions

```
public class EventHandlingWithLambda {  
    public static void main(String[] args) {  
        // Create an instance of the custom event source  
        CustomEventSource eventSource = new  
CustomEventSource();  
  
        // Register a Lambda expression as an event handler  
        eventSource.setEventHandler(() -> {  
            System.out.println("Custom Event Triggered using  
Lambda!");  
        });  
  
        // Simulate the occurrence of a custom event  
        eventSource.triggerEvent();  
    }  
}  
  
@FunctionalInterface  
interface EventHandler {  
    void handleEvent();  
}  
  
class CustomEventSource {  
    private EventHandler eventHandler;  
  
    public void setEventHandler(EventHandler eventHandler) {  
        this.eventHandler = eventHandler;  
    }  
}
```

```

public void triggerEvent() {
    if (eventHandler != null) {
        eventHandler.handleEvent();
    }
}
}

```



```

Custom Event Triggered using Lambda!

```

4. Write a Java program for Iterating over List using Lambda expressions

```

import java.util.ArrayList;
import java.util.List;


```

```

public class IterateListWithLambda {
    public static void main(String[] args) {
        // Create a list of strings
        List<String> stringList = new ArrayList<>();
        stringList.add("Apple");
        stringList.add("Banana");
        stringList.add("Cherry");
        stringList.add("Date");

        // Iterate over the list using Lambda expression
        stringList.forEach(item -> System.out.println(item));
    }
}

```



```
Apple
Banana
Cherry
Date
```

5. Write a Java program to combine Predicate in Lambda Expressions

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import java.util.function.Predicate;
```

```
class Person {
```

```
    private String name;
```

```
    private int age;
```

```
    public Person(String name, int age) {
```

```
        this.name = name;
```

```
        this.age = age;
```

```
    }
```

```
    public String getName() {
```

```
        return name;
```

```
    }
```

```
    public int getAge() {
```

```
        return age;
```

```
    }
```

```
}
```

```
public class CombinePredicatesWithLambda {
```

```
    public static void main(String[] args) {
```

```
        // Create a list of Person objects
```

```
        List<Person> people = new ArrayList<>();
```

```
        people.add(new Person("Alice", 25));
```

```
        people.add(new Person("Bob", 30));
```

```
        people.add(new Person("Charlie", 20));
```

```
        people.add(new Person("David", 35));
```

```

        // Create Predicate objects for filtering
        Predicate<Person> agePredicate = person -> person.getAge()
        >= 30;
        Predicate<Person> namePredicate = person ->
        person.getName().startsWith("B");

        // Combine the predicates using Lambda expressions
        Predicate<Person> combinedPredicate =
        agePredicate.and(namePredicate);

        // Filter the list of people based on the combined predicate
        List<Person> filteredPeople = filterPeople(people,
        combinedPredicate);

        // Print the filtered list
        for (Person person : filteredPeople) {
            System.out.println(person.getName() + " - " +
        person.getAge());
        }
    }

    public static List<Person> filterPeople(List<Person> people,
    Predicate<Person> predicate) {
        List<Person> filteredPeople = new ArrayList<>();
        for (Person person : people) {
            if (predicate.test(person)) {
                filteredPeople.add(person);
            }
        }
        return filteredPeople;
    }
}

```


RESULT

Bob - 30

6. Write a Java program for creating a List of String by filtering

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import java.util.stream.Collectors;
```

```
public class FilterListExample {
```

```
    public static void main(String[] args) {
```

```
        // Create a list of strings
```

```
        List<String> originalList = new ArrayList<>();
```

```
        originalList.add("Apple");
```

```
        originalList.add("Banana");
```

```
        originalList.add("Cherry");
```

```
        originalList.add("Date");
```

```
        originalList.add("Fig");
```

```
        originalList.add("Grape");
```

```
        // Create a filtered list of strings using Java 8 Streams
```

```
        List<String> filteredList = originalList.stream()
```

```
            .filter(s -> s.startsWith("A") || s.startsWith("C")) // Filter  
based on criteria (e.g., starts with "A" or "C")
```

```
            .collect(Collectors.toList());
```

```
        // Print the filtered list
```

```
        System.out.println("Filtered List:");
```

```
        for (String item : filteredList) {
```

```
            System.out.println(item);
```

```
        }
```

```
    }
```

```
}
```

RESULT

```
Filtered List:Starts with A or C  
Apple  
Cherry
```

7. Write a Java program for creating a Sub List by Copying distinct values

```
import java.util.ArrayList;  
import java.util.List;
```

```
public class CopyDistinctSubListExample {  
    public static void main(String[] args) {  
        // Create a list of integers with duplicate values  
        List<Integer> originalList = new ArrayList<>();  
        originalList.add(1);  
        originalList.add(2);  
        originalList.add(2);  
        originalList.add(3);  
        originalList.add(4);  
        originalList.add(4);  
        originalList.add(5);  
  
        // Create a new list to store distinct values  
        List<Integer> distinctList = new ArrayList<>();  
  
        // Copy distinct values to the new list  
        for (Integer value : originalList) {  
            if (!distinctList.contains(value)) {  
                distinctList.add(value);  
            }  
        }  
  
        // Print the distinct sub-list  
        System.out.println("Distinct Sub-List:");  
        for (Integer value : distinctList) {  
            System.out.println(value);  
        }  
    }  
}
```

RESULT

Distinct Sub-List:

- 1
- 2
- 3
- 4
- 5