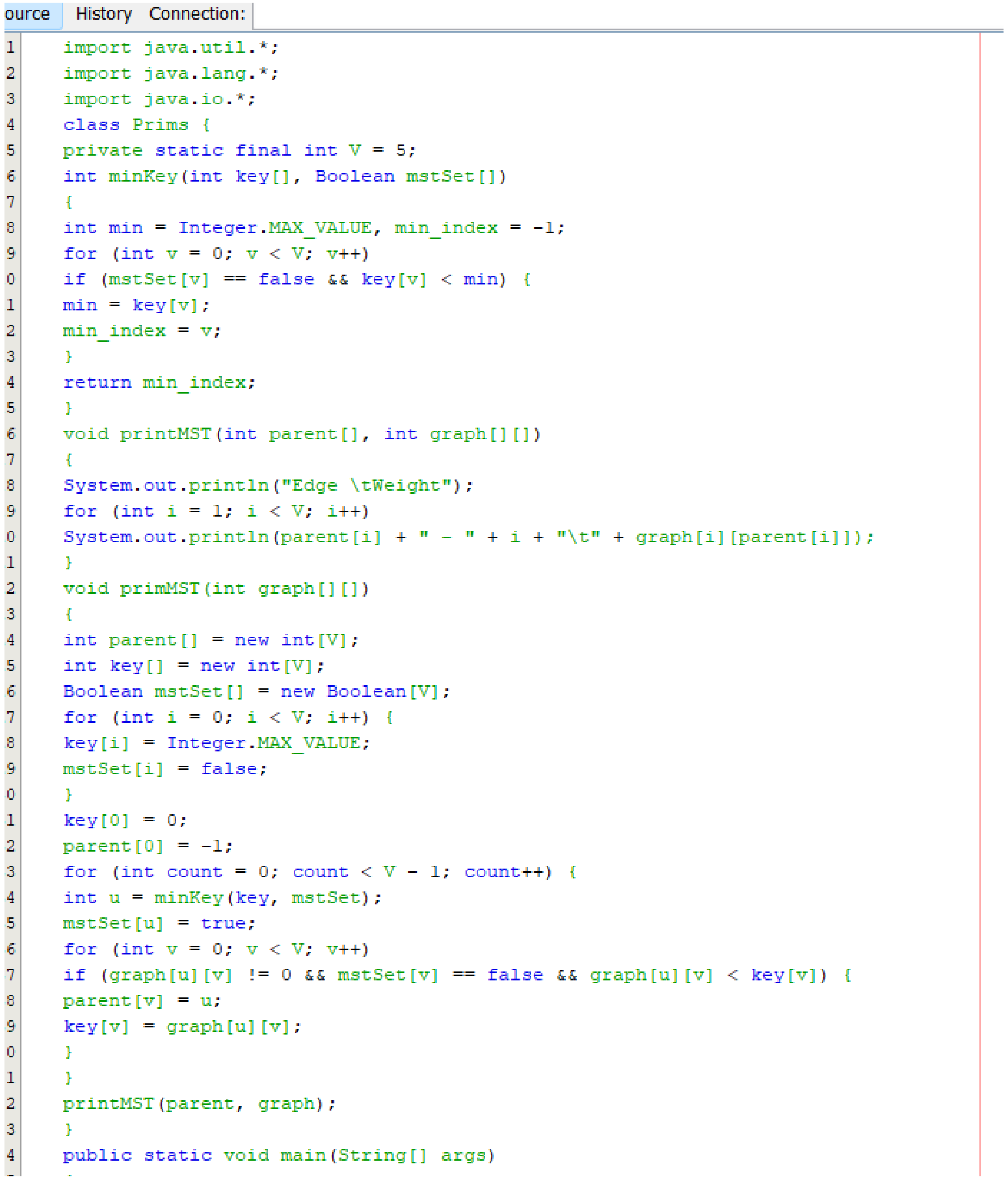
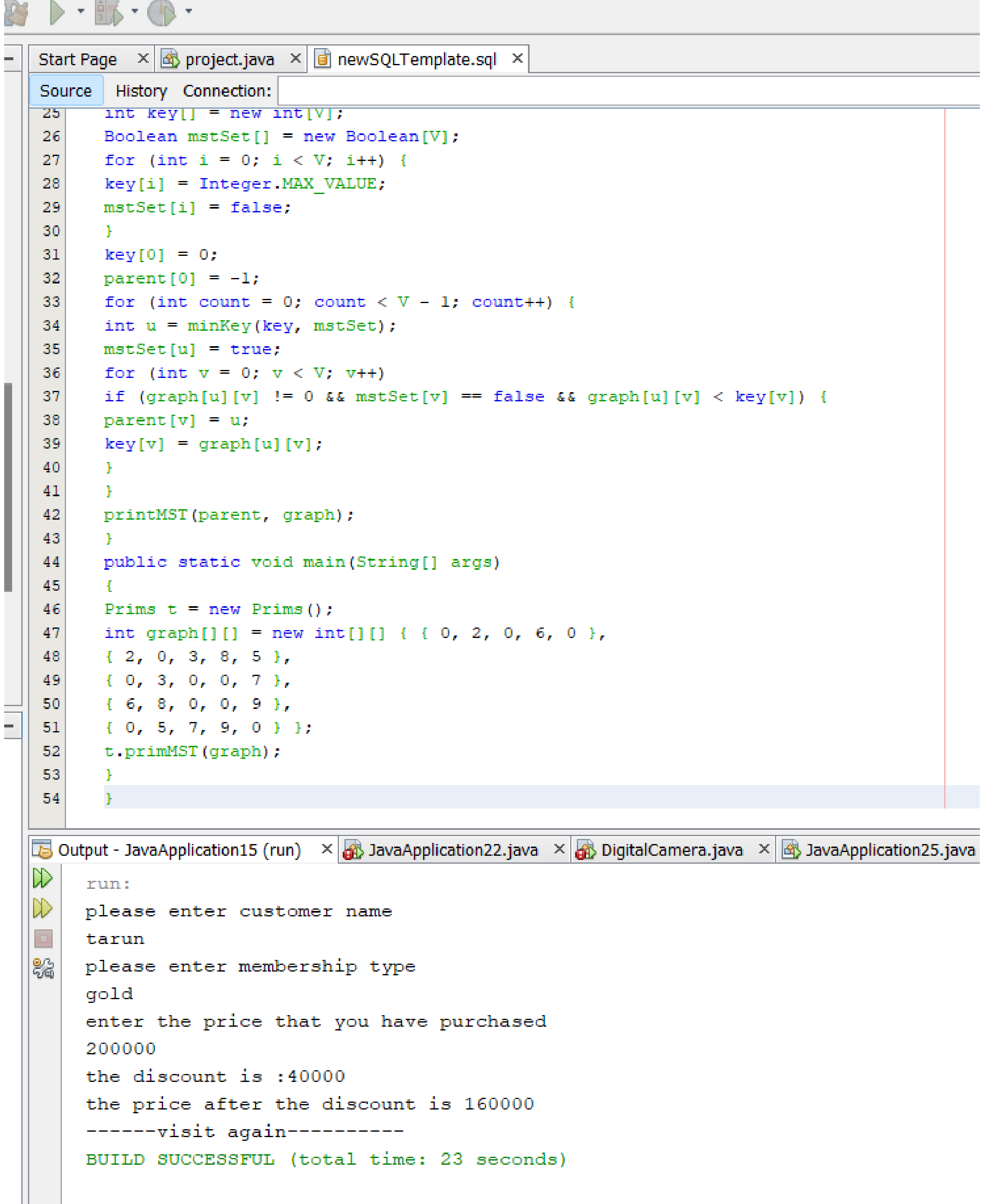
TARUN

19BCE7578

CSE3004



```
import java.util.*;
import java.lang.*;
import java.io.*;
class Prims {
private static final int V = 5;
int minKey(int key[], Boolean mstSet[])
{
int min = Integer.MAX_VALUE, min_index = -1;
for (int v = 0; v < V; v++)
if (mstSet[v] == false && key[v] < min) {
min = key[v];
min_index = v;
}
return min_index;
}
void printMST(int parent[], int graph[][])
{
System.out.println("Edge \tWeight");
for (int i = 1; i < V; i++)
System.out.println(parent[i] + " - " + i + "\t" + graph[i][parent[i]]);
}
void primMST(int graph[][])
{
int parent[] = new int[V];
int key[] = new int[V];
Boolean mstSet[] = new Boolean[V];
for (int i = 0; i < V; i++) {
key[i] = Integer.MAX_VALUE;
mstSet[i] = false;
}
key[0] = 0;
parent[0] = -1;
for (int count = 0; count < V - 1; count++) {
int u = minKey(key, mstSet);
mstSet[u] = true;
for (int v = 0; v < V; v++)
if (graph[u][v] != 0 && mstSet[v] == false && graph[u][v] < key[v]) {
parent[v] = u;
key[v] = graph[u][v];
}
}
printMST(parent, graph);
}
public static void main(String[] args)
```



```java
int key[] = new int[V];
Boolean mstSet[] = new Boolean[V];
for (int i = 0; i < V; i++) {
key[i] = Integer.MAX_VALUE;
mstSet[i] = false;
}
key[0] = 0;
parent[0] = -1;
for (int count = 0; count < V - 1; count++) {
int u = minKey(key, mstSet);
mstSet[u] = true;
for (int v = 0; v < V; v++)
if (graph[u][v] != 0 && mstSet[v] == false && graph[u][v] < key[v]) {
parent[v] = u;
key[v] = graph[u][v];
}
}
printMST(parent, graph);
}
public static void main(String[] args)
{
Prims t = new Prims();
int graph[][] = new int[][] { { 0, 2, 0, 6, 0 },
{ 2, 0, 3, 8, 5 },
{ 0, 3, 0, 0, 7 },
{ 6, 8, 0, 0, 9 },
{ 0, 5, 7, 9, 0 } };
t.primMST(graph);
}
}
```

```
run:
please enter customer name
tarun
please enter membership type
gold
enter the price that you have purchased
200000
the discount is :40000
the price after the discount is 160000
------visit again----------
BUILD SUCCESSFUL (total time: 23 seconds)
```

Time complexity

Worst case

Using binary heap→ $O(Elogv)$

Using Fibonacci heap→$O(E+ vlogv)$

Overall time complexity

$=>O(E=v)xO(logv)$

$=>O((E+v)logv)$

$=>O(Elogv)$