Bajrang 11212766

BCSE-506L (Performance Analysis of Programming Languages Lab)

<u>EXPERIMENT NO. 8A1</u>

AIM: WP based upon the rule you learn for CONWAY'S game of life using c/python and java.
Overview:
Initially, there is a grid with some cells which may be alive or dead. Our task is to generate the next generation of cells based on the following rules:

1. Any live cell with fewer than two live neighbors dies as if caused by underpopulation.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

<u>Overview by WikiPedia</u>

Highlight: Used **throw** keyword,Used **for** loop and **if else.**

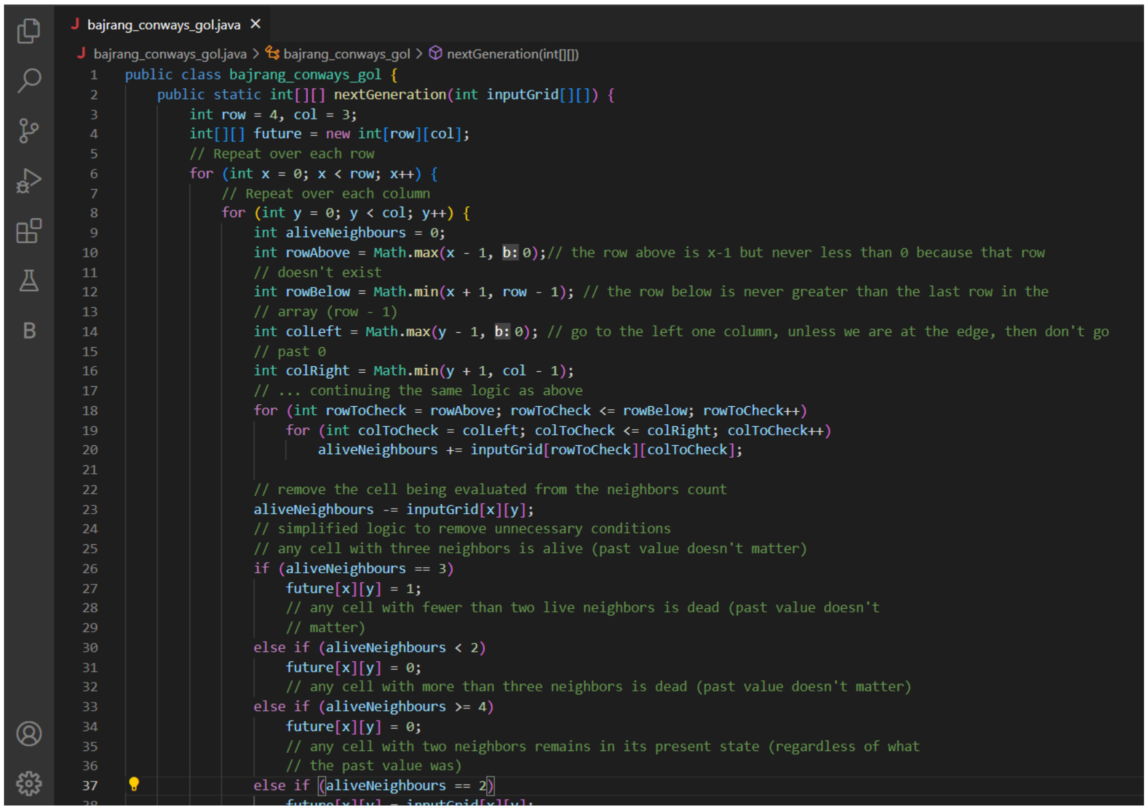
Java Program:

Source Code:
```
public class bajrang_conways_gol {
    public static int[][] nextGeneration(int inputGrid[][]) {
        int row = 4, col = 3;
        int[][] future = new int[row][col];
        for (int x = 0; x < row; x++) {
            for (int y = 0; y < col; y++) {
                int aliveNeighbours = 0;
                int rowAbove = Math.max(x - 1, 0);
                int rowBelow = Math.min(x + 1, row - 1);
                int colLeft = Math.max(y - 1, 0);
                int colRight = Math.min(y + 1, col - 1);
                for (int rowToCheck = rowAbove; rowToCheck <=
rowBelow; rowToCheck++)
                    for (int colToCheck = colLeft; colToCheck <=
colRight; colToCheck++)
                        aliveNeighbours +=
inputGrid[rowToCheck][colToCheck];
                aliveNeighbours -= inputGrid[x][y];
                if (aliveNeighbours == 3)
                    future[x][y] = 1;
                else if (aliveNeighbours < 2)
                    future[x][y] = 0;

                else if (aliveNeighbours >= 4)
                    future[x][y] = 0;

                else if (aliveNeighbours == 2)
                    future[x][y] = inputGrid[x][y];
                else
                    throw new RuntimeException("Unhandled neighbor
condition");
            }
        } return future;
    }
    public static void main(String[] args) {
        int[][] game = { { 0, 1, 0 },
                { 0, 0, 1 },
                { 1, 1, 1 },
                { 0, 0, 0 }
        };
        var nextGen = nextGeneration(game);
        for (var row : nextGen) {
            for (var cell : row) {
                System.out.print(cell);
            }
            System.out.println();
        }
    }
}
```

OUTPUT:

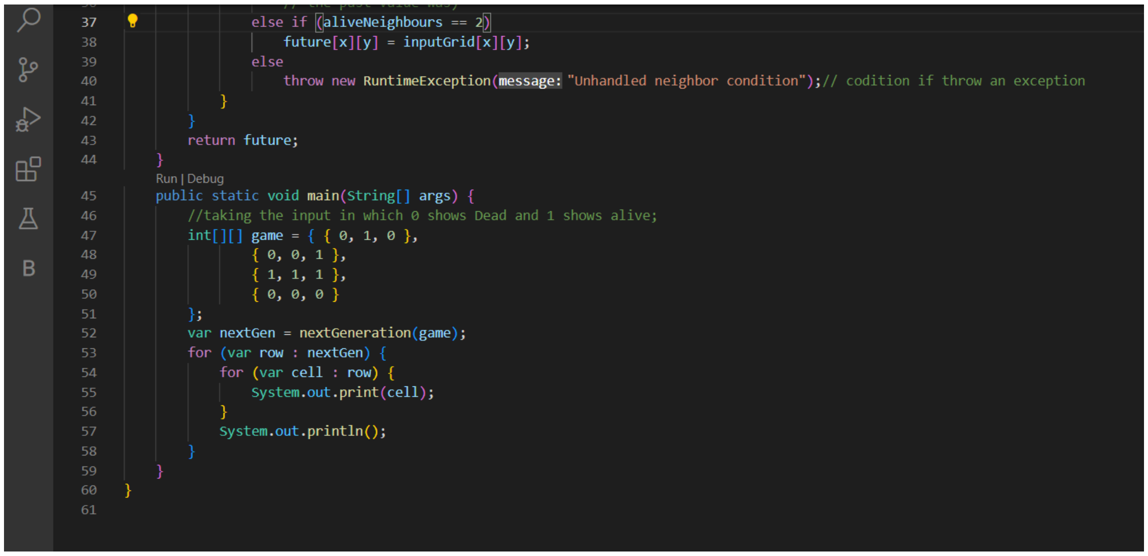
```
PS E:\Github\conways_game_of_life>  e:; cd
'e:\Github\conways_game_of_life'; & 'C:\Program
Files\Java\jdk-19\bin\java.exe' '--enable-preview'\bin'
'bajrang_conways_gol'
000
101
011
010
PS E:\Github\conways_game_of_life>
```

IDE INPUT & Output:

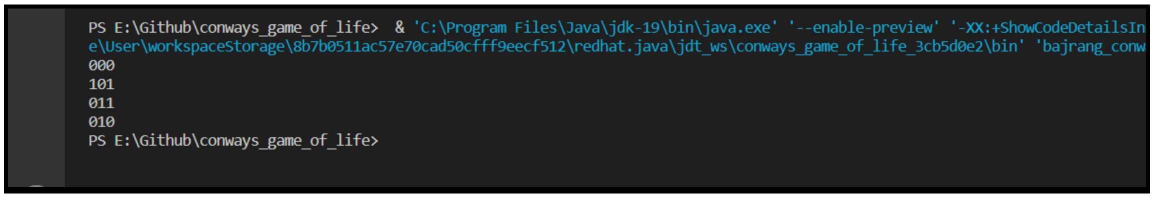


bajrang_conways_gol.java
bajrang_conways_gol.java > bajrang_conways_gol > nextGeneration(int[][])

```java
public class bajrang_conways_gol {
    public static int[][] nextGeneration(int inputGrid[][]) {
        int row = 4, col = 3;
        int[][] future = new int[row][col];
        // Repeat over each row
        for (int x = 0; x < row; x++) {
            // Repeat over each column
            for (int y = 0; y < col; y++) {
                int aliveNeighbours = 0;
                int rowAbove = Math.max(x - 1, b: 0);// the row above is x-1 but never less than 0 because that row
                // doesn't exist
                int rowBelow = Math.min(x + 1, row - 1); // the row below is never greater than the last row in the
                // array (row - 1)
                int colLeft = Math.max(y - 1, b: 0); // go to the left one column, unless we are at the edge, then don't go
                // past 0
                int colRight = Math.min(y + 1, col - 1);
                // ... continuing the same logic as above
                for (int rowToCheck = rowAbove; rowToCheck <= rowBelow; rowToCheck++)
                    for (int colToCheck = colLeft; colToCheck <= colRight; colToCheck++)
                        aliveNeighbours += inputGrid[rowToCheck][colToCheck];

                // remove the cell being evaluated from the neighbors count
                aliveNeighbours -= inputGrid[x][y];
                // simplified logic to remove unnecessary conditions
                // any cell with three neighbors is alive (past value doesn't matter)
                if (aliveNeighbours == 3)
                    future[x][y] = 1;
                    // any cell with fewer than two live neighbors is dead (past value doesn't
                    // matter)
                else if (aliveNeighbours < 2)
                    future[x][y] = 0;
                    // any cell with more than three neighbors is dead (past value doesn't matter)
                else if (aliveNeighbours >= 4)
                    future[x][y] = 0;
                    // any cell with two neighbors remains in its present state (regardless of what
                    // the past value was)
                else if (aliveNeighbours == 2)
```



```java
                else if (aliveNeighbours == 2)
                    future[x][y] = inputGrid[x][y];
                else
                    throw new RuntimeException(message: "Unhandled neighbor condition");// codition if throw an exception
            }
        }
        return future;
    }
    Run | Debug
    public static void main(String[] args) {
        //taking the input in which 0 shows Dead and 1 shows alive;
        int[][] game = { { 0, 1, 0 },
                { 0, 0, 1 },
                { 1, 1, 1 },
                { 0, 0, 0 }
        };
        var nextGen = nextGeneration(game);
        for (var row : nextGen) {
            for (var cell : row) {
                System.out.print(cell);
            }
            System.out.println();
        }
    }
}
```



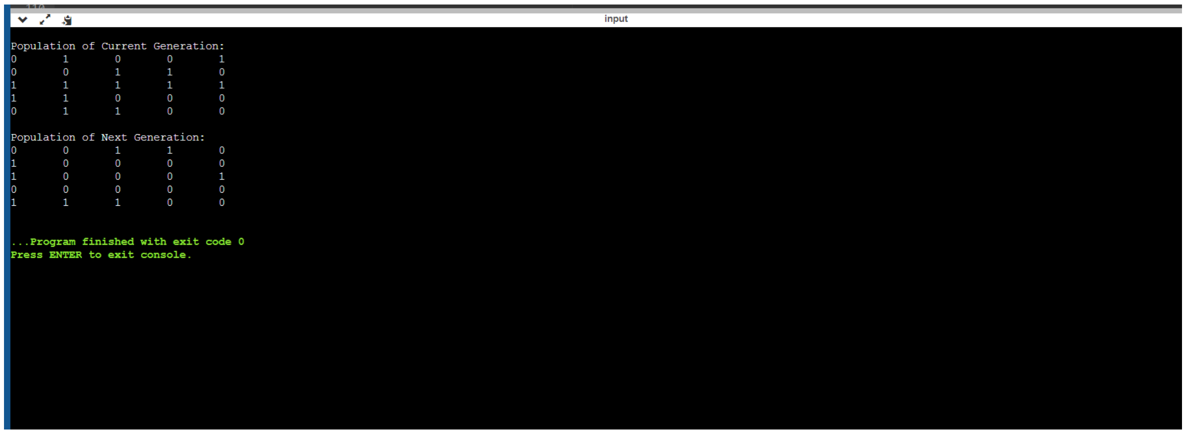
```
PS E:\Github\conways_game_of_life>  & 'C:\Program Files\Java\jdk-19\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsIn
e\User\workspaceStorage\8b7b0511ac57e70cad50cfff9eecf512\redhat.java\jdt_ws\conways_game_of_life_3cb5d0e2\bin' 'bajrang_conw
000
101
011
010
PS E:\Github\conways_game_of_life>
```

C Program

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define ROW 5
#define COL 5
int randomnumber=20;
int Current_Gen[ROW+2][COL+2];
int Next_Gen[ROW+2][COL+2];
int Alive_Neighbours[ROW+2][COL+2];
int main()
{
int i,j,a,b,sum;
for(i=0;i<ROW+2;i++)
for(j=0;j<COL+2;j++)
Current_Gen[i][j]=0;
for(i=1;i<=randomnumber;i++)
{
a = 1+(rand() % ROW);
b = 1+(rand() % COL);
Current_Gen[a][b]=1;
}
printf("\nPopulation of Current Generation:\n");
for(i=1;i<ROW+1;i++)
{
for(j=1;j<COL+1;j++)
printf("%d\t", Current_Gen[i][j]);
printf("\n");
}
//Calculating Alive neighbours for each node
for(i=1;i<ROW+1;i++)
{
for(j=1;j<COL+1;j++)
{
sum=0;
sum=sum+Current_Gen[i-1][j-1];
sum=sum+Current_Gen[i-1][j];
sum=sum+Current_Gen[i-1][j+1];
sum=sum+Current_Gen[i][j-1];
sum=sum+Current_Gen[i][j+1];

sum=sum+Current_Gen[i+1][j-1];
sum=sum+Current_Gen[i+1][j];
sum=sum+Current_Gen[i+1][j+1];
Alive_Neighbours[i][j]=sum;
if(Current_Gen[i][j]==1)
{
if((Alive_Neighbours[i][j]<2)||(Alive_Neighbours[i][j]>3))
Next_Gen[i][j]=0;
else
Next_Gen[i][j]=1;
}
else
{
if(Alive_Neighbours[i][j]==3)
Next_Gen[i][j]=1;
else
Next_Gen[i][j]=0;
}
}
}
printf("\nPopulation of Next Generation:\n");
for(i=1;i<ROW+1;i++)
{
for(j=1;j<COL+1;j++)
printf("%d\t", Next_Gen[i][j]);
printf("\n");
}
return 0;
}
```

Output:



```
Population of Current Generation:
0       1       0       0       1
0       0       1       1       0
1       1       1       1       1
1       1       0       0       0
0       1       1       0       0

Population of Next Generation:
0       0       1       1       0
1       0       0       0       0
1       0       0       0       1
0       0       0       0       0
1       1       1       0       0


...Program finished with exit code 0
Press ENTER to exit console.
```

Bajrang Gour

Roll No 11212766

Section- B3