Practical No 04

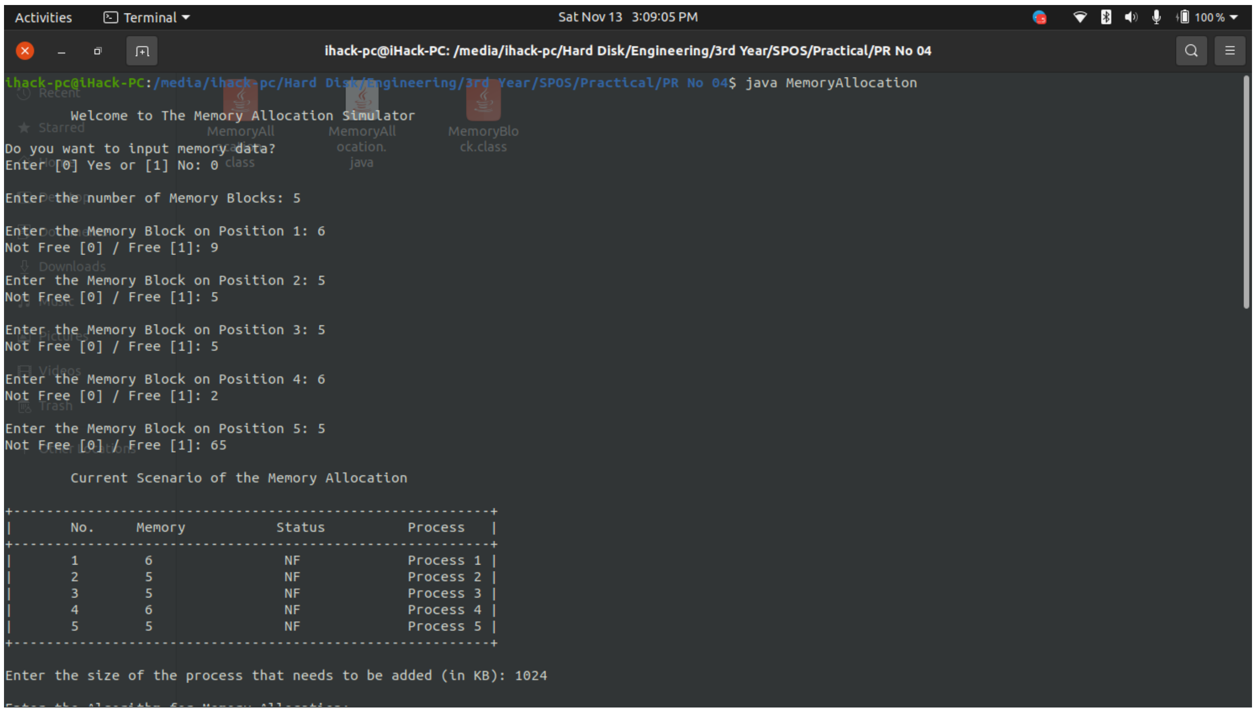
Title:- Write a program to simulate memory replacement strategies- First Fit, Best Fit, Worst Fit.

Source Code :-

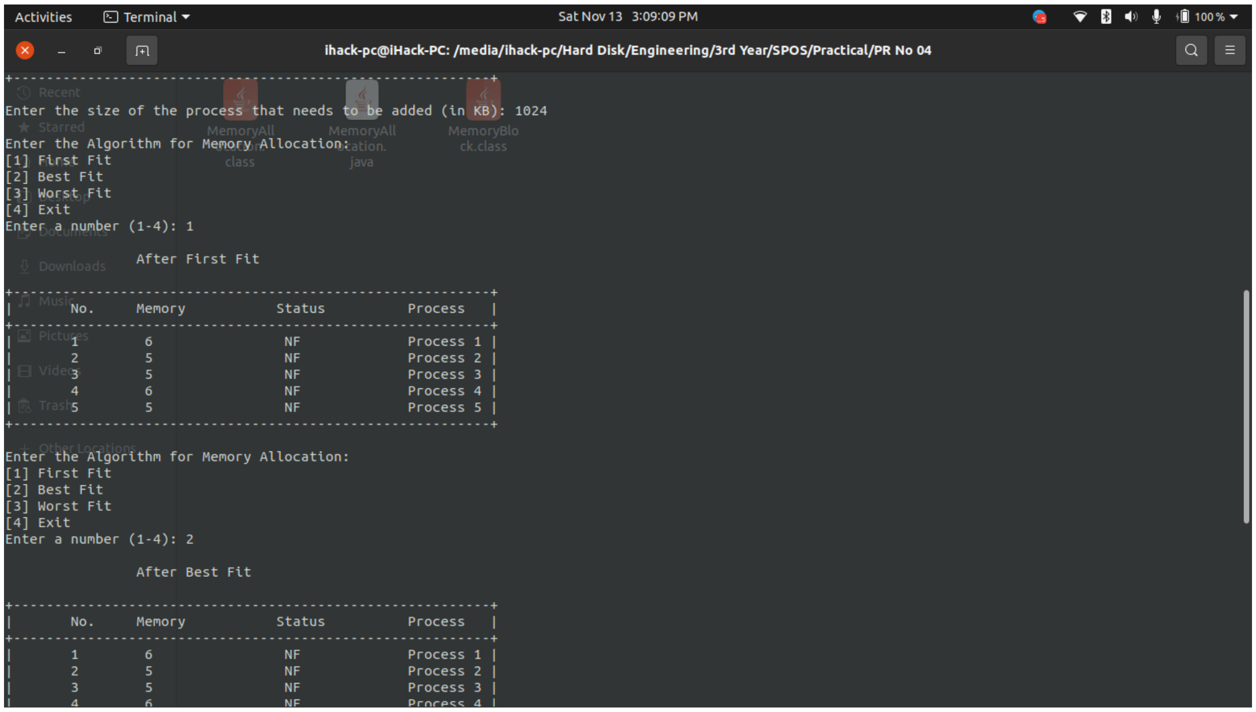
```
import java.io.*;
import java.util.*;
class MemoryBlock {
  // hardcoded values if the user does not want to enter the values;
  int[] memory = new int[] { 100, 300, 40, 50, 150, 240, 200, 400};
  boolean[] free = new boolean[] { false, true, false, true, false, true, false, true };
  int processNumber = 4;
  int divs = memory.length;
  int processSize;
  Scanner s = new Scanner(System.in);

  void welcomeMessage() {
    System.out.println("\n\tWelcome to The Memory Allocation Simulator");

    System.out.print("\nDo you want to input memory data? \nEnter [0] Yes or [1] No: ");
    int inputData = s.nextInt();
    if(inputData == 0)
      memoryInput();
    else
      processInput();
  }

  void processInput() {
    System.out.println("\n\tCurrent Scenario of the Memory Allocation \n");
    printTable(-1);
    System.out.print("\nEnter the size of the process that needs to be added (in KB): ");
    processSize = s.nextInt();
    choice();
  }

  void memoryInput() {
    // re-initialising the data if user wants to enter the data;
    memory = new int[100];
    free = new boolean[100];
    processNumber = 0;
    System.out.print("\nEnter the number of Memory Blocks: ");
    divs = s.nextInt();
    for(int i = 0; i < divs; ++i) {
        System.out.print("\nEnter the Memory Block on Position " + (i + 1) + ": ");
        memory[i] = s.nextInt();
        System.out.print("Not Free [0] / Free [1]: ");
        free[i] = ((s.nextInt() == 1) ? true : false);
        if(!free[i]) {
            processNumber += 1;
        }
    }
    processInput();
}

void choice() {
    boolean running = true;
    while(running) {
        System.out.print("\nEnter the Algorithm for Memory Allocation: \n");
        System.out.print("[1] First Fit\n");
        System.out.print("[2] Best Fit\n");
        System.out.print("[3] Worst Fit\n");
        System.out.print("[4] Exit\n");

        System.out.print("Enter a number (1-4): ");
        int fitType = s.nextInt();
        switch(fitType) {
            case 1:
                System.out.println("\n\t\tAfter First Fit \n");
                firstFit();
                break;
            case 2:
                System.out.println("\n\t\tAfter Best Fit  \n");
                bestFit();
                break;
            case 3:
                System.out.println("\n\t\tAfter Worst Fit \n");
                worstFit();
                break;
            case 4:
                running = false;
                break;
            default:
                System.out.println("\nPlease enter a number between 1 and 4.\n");
        }
    }
}
```

```
void firstFit() {
    int ans = -1;
    for(int i = 0; i < divs; i++) {
        if(free[i] && processSize <= memory[i]) {
            ans = i;
            break;
        }
    }
    printTable(ans);
}

void bestFit() {
    int ans = -1, curr = 1000000;
    for(int i = 0; i < divs; i++) {
        if(free[i] && processSize <= memory[i]) {
            if(memory[i] - processSize < curr) {
                curr = memory[i] - processSize;
                ans = i;
            }
        }
    }
    printTable(ans);
}

void worstFit() {
    int ans = -1, curr = 0;
    for(int i = 0; i < divs; i++) {
        if(free[i] && processSize <= memory[i]) {
            if(memory[i] - processSize > curr) {
                curr = memory[i] - processSize;
                ans = i;
            }
        }
    }
    printTable(ans);
}

void printTable(int pos) {
    System.out.print("+------------------------------------------------------------+\n");
    System.out.print("|\tNo.\tMemory \t\t Status \t Process   |\n");
    System.out.print("+------------------------------------------------------------+\n");
    int j = 1, ok = 0;
    for (int i = 0; i < divs; i++) {
            if(i == pos) {
                System.out.print("|\t" + (i + 1) + " \t " +  processSize + "  \t\t " + " NF \t\t " + "Process "
+ (processNumber + 1) + " |");
                if(memory[i] - processSize != 0) {
                    System.out.print("\n|\t" + (i + 2) + " \t " + (memory[i] - processSize) + "  \t\t " + " F
\t\t\t   |");
                    ok = 1;
                }
            }
            else {
                System.out.print("|\t" + (i + 1 + ok) + " \t " +  memory[i] + "  \t\t  " + ((free[i]) ? "F \t\t\t   |"
: "NF \t\t " + "Process " + j++ + " |"));
            }
            System.out.println(' ');
        }
        System.out.print("+---------------------------------------------------------+\n");
    }
}

class MemoryAllocation {
    public static void main(String args[]) throws IOException
    {
        MemoryBlock m = new MemoryBlock();
        m.welcomeMessage();
    }
}
```

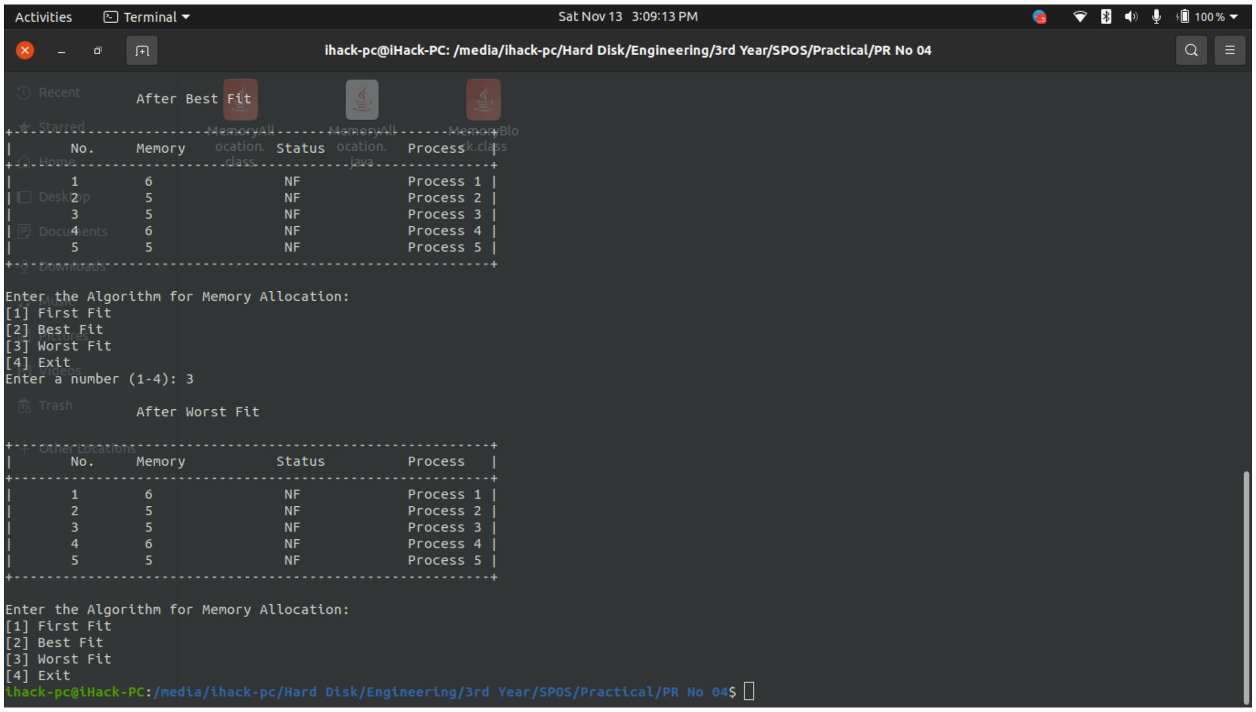
Output :-



```
ihack-pc@iHack-PC:/media/ihack-pc/Hard Disk/Engineering/3rd Year/SPOS/Practical/PR No 04$ java MemoryAllocation

        Welcome to The Memory Allocation Simulator

Do you want to input memory data?
Enter [0] Yes or [1] No: 0

Enter the number of Memory Blocks: 5

Enter the Memory Block on Position 1: 6
Not Free [0] / Free [1]: 9

Enter the Memory Block on Position 2: 5
Not Free [0] / Free [1]: 5

Enter the Memory Block on Position 3: 5
Not Free [0] / Free [1]: 5

Enter the Memory Block on Position 4: 6
Not Free [0] / Free [1]: 2

Enter the Memory Block on Position 5: 5
Not Free [0] / Free [1]: 65

        Current Scenario of the Memory Allocation

+------------------------------------------------------------+
|       No.     Memory           Status          Process    |
+------------------------------------------------------------+
|       1        6                NF             Process 1 |
|       2        5                NF             Process 2 |
|       3        5                NF             Process 3 |
|       4        6                NF             Process 4 |
|       5        5                NF             Process 5 |
+------------------------------------------------------------+

Enter the size of the process that needs to be added (in KB): 1024
```



```
+------------------------------------------------------------+
Enter the size of the process that needs to be added (in KB): 1024

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
Enter a number (1-4): 1

                After First Fit

+------------------------------------------------------------+
|       No.     Memory           Status          Process    |
+------------------------------------------------------------+
|       1        6                NF             Process 1 |
|       2        5                NF             Process 2 |
|       3        5                NF             Process 3 |
|       4        6                NF             Process 4 |
|       5        5                NF             Process 5 |
+------------------------------------------------------------+

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
Enter a number (1-4): 2

                After Best Fit

+------------------------------------------------------------+
|       No.     Memory           Status          Process    |
+------------------------------------------------------------+
|       1        6                NF             Process 1 |
|       2        5                NF             Process 2 |
|       3        5                NF             Process 3 |
|       4        6                NF             Process 4 |
```


Activities
Terminal
Sat Nov 13 3:09:13 PM
100 %
ihack-pc@iHack-PC: /media/ihack-pc/Hard Disk/Engineering/3rd Year/SPOS/Practical/PR No 04

```
                After Best Fit

+------------------------------------------------------------+
|       No.     Memory           Status          Process     |
+------------------------------------------------------------+
|       1       6                 NF             Process 1 |
|       2       5                 NF             Process 2 |
|       3       5                 NF             Process 3 |
|       4       6                 NF             Process 4 |
|       5       5                 NF             Process 5 |
+------------------------------------------------------------+

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
Enter a number (1-4): 3
                After Worst Fit

+------------------------------------------------------------+
|       No.     Memory           Status          Process     |
+------------------------------------------------------------+
|       1       6                 NF             Process 1 |
|       2       5                 NF             Process 2 |
|       3       5                 NF             Process 3 |
|       4       6                 NF             Process 4 |
|       5       5                 NF             Process 5 |
+------------------------------------------------------------+

Enter the Algorithm for Memory Allocation:
[1] First Fit
[2] Best Fit
[3] Worst Fit
[4] Exit
ihack-pc@iHack-PC:/media/ihack-pc/Hard Disk/Engineering/3rd Year/SPOS/Practical/PR No 04$ 
```