```
/*****************************************************************************
Write a Java program to implement following memory allocation
strategies: First Fit, Best Fit and Worst Fit.
*****************************************************************************/
import java.util.*;
public class Main
{
   //first fit:allocates the first hole that is big enough
   static void firstFit(int hole[],int ps[],int flag[])
   {
      int c1;
      for(int j=0;j<ps.length;j++)
      {
         c1=0;
        for(int i=0;i<hole.length;i++)
        {
         if(hole[i]>=ps[j] && flag[i]==0)
         {
            c1=1;
           flag[i]=1;
           System.out.println("\nProcess of size "+ps[j]+" is allocated to memory block of size "+hole[i]);
           break;
         }
        }
        if(c1==0)
        {
           System.out.println("\nProcess of size "+ps[j]+" could not be allocated a memory block");
        }
     }
   }

//best fit:allocates the smallest hole that is big enough
  static void BestFit(int holes[],int ps[],int flag[])
  {
     int temp,n1,c1=0;
     n1=holes.length;
     for(int k=0;k<flag.length;k++)
     {
        flag[k]=0;
     }
      //arrange holes in ascending order using selection sort
     for (int i = 0; i < n1-1; i++)
      {
         // Find the minimum element in unsorted array
         int min_idx = i;
         for (int j = i+1; j < n1; j++)
            if (holes[j] < holes[min_idx])
               min_idx = j;

         // Swap the found minimum element with the first
         // element
         temp = holes[min_idx];
         holes[min_idx] = holes[i];
         holes[i] = temp;
```

```
    }

    for(int i=0;i<holes.length;i++)
    {
       System.out.print(holes[i]+" ");
    }

    int i;
   for(int j=0;j<ps.length;j++)
   {
      c1=0;
     for(i=0;i<holes.length;i++)
     {

      if(holes[i]>=ps[j]&& flag[i]==0)
      {
         c1=1;
         flag[i]=1;
         System.out.println("\nProcess of size "+ps[j]+" is allocated to memory block of size "+holes[i]);
         break;

      }
     }
     if(c1==0)
     {
        System.out.println("\nProcess of size "+ps[j]+" could not be allocated a memory block");
     }
   }
  }
//worst fit:allocates the largest hole :must search the entire list and produces the largest left over hole
  static void WorstFit(int holes[],int ps[],int flag[])
  {
    //arrange holes in descending order using selection sort

    int maxid,temp,c2;
    for(int k=0;k<flag.length;k++)
    {
       flag[k]=0;
    }
    for(int i=0;i<holes.length-1;i++)
    {
       maxid=i;
       for(int j=i+1;j<holes.length;j++)
       {
         if(holes[j]>holes[maxid])
         {
            maxid=j;
         }

       }
       temp=holes[maxid];
       holes[maxid]=holes[i];
       holes[i]=temp;
    }
```

```
      for(int i=0;i<holes.length;i++)
     {
        System.out.print(holes[i]+" ");
     }

     int i;
     for(int j=0;j<ps.length;j++)
     {
        c2=0;
      for(i=0;i<holes.length;i++)
      {
       if(holes[i]>=ps[j]&& flag[i]==0)
       {
          c2=1;
          flag[i]=1;
          System.out.println("\nProcess of size "+ps[j]+" is allocated to memory block of size "+holes[i]);
          break;

       }
      }
      if(c2==0)
      {
         System.out.println("\nProcess of size "+ps[j]+" could not be allocated a memory block");
      }
    }
  }

public static void main(String args[])
{
      Scanner sc=new Scanner(System.in);

      int ch,n,p;
      char cont;
      System.out.println("\nEnter no. of memory blocks available :");
      n=sc.nextInt();

      int hole[]=new int[n];
      int mid[]=new int[n];
      int flag[]=new int[n];

      for(int i=0;i<n;i++)
      {
       System.out.println("\nEnter the block id :");
       mid[i]=sc.nextInt();
       System.out.println("\nEnter the size of memory block in KB :");
       hole[i]=sc.nextInt();
       flag[i]=0;
      }

      System.out.println("\nEnter no. of processes available :");
      p=sc.nextInt();

      /*MemmoryAllocation t=new MemmoryAllocation(n,p);
```

```
        MemmoryAllocation t1=new MemmoryAllocation(n,p);
        MemmoryAllocation t2=new MemmoryAllocation(n,p);*/
        int pobj[]=new int[p];
        int pid[]=new int[p];


        for(int i=0;i<p;i++)
        {
         System.out.println("\nEnter the process id:");
         pid[i]=sc.nextInt();
         System.out.println("\nEnter the size of process:");
         pobj[i]=sc.nextInt();
          flag[i]=0;
        }


        do
        {
         System.out.println("\n****MENU******");
         System.out.println("\n1]FIRST FIT STRATEGY");
         System.out.println("\n2]BEST FIT STRATEGY");
         System.out.println("\n3]WORST FIT STRATEGY");
         System.out.println("\n4]EXIT");

         System.out.println("\nEnter your option :");
         ch=sc.nextInt();

         switch(ch)
         {
            case 1:firstFit(hole,pobj,flag);
                   break;

            case 2:BestFit(hole,pobj,flag);
                   break;

            case 3:WorstFit(hole,pobj,flag);
                   break;

            case 4:System.out.println("\nThank you!");
                   break;
            default: System.out.println("\nInvalid choice!");
         }
       System.out.println("\nDo you want to continue?(y/n)");
       cont=sc.next().charAt(0);
       }while(cont=='y'||cont=='Y');
       System.out.println("\nEXECUTION COMPLETED!!");
    }
}

/*OUTPUT

Enter no. of memory blocks available :
5
```

Enter the block id :
1

Enter the size of memory block in KB :
100

Enter the block id :
2

Enter the size of memory block in KB :
500

Enter the block id :
3

Enter the size of memory block in KB :
200

Enter the block id :
4

Enter the size of memory block in KB :
300

Enter the block id :
5

Enter the size of memory block in KB :
600

Enter no. of processes available :
4

Enter the process id:
1

Enter the size of process:
212

Enter the process id:
2

Enter the size of process:
417

Enter the process id:
3

Enter the size of process:
112

Enter the process id:
4

Enter the size of process:
426

****MENU******

1]FIRST FIT STRATEGY

2]BEST FIT STRATEGY

3]WORST FIT STRATEGY

4]EXIT

Enter your option :
1

Process of size 212 is allocated to memory block of size 500

Process of size 417 is allocated to memory block of size 600

Process of size 112 is allocated to memory block of size 200

Process of size 426 could not be allocated a memory block

Do you want to continue?(y/n)
y

****MENU******

1]FIRST FIT STRATEGY

2]BEST FIT STRATEGY

3]WORST FIT STRATEGY

4]EXIT

Enter your option :
2
100 200 300 500 600
Process of size 212 is allocated to memory block of size 300

Process of size 417 is allocated to memory block of size 500

Process of size 112 is allocated to memory block of size 200

Process of size 426 is allocated to memory block of size 600

Do you want to continue?(y/n)
y

****MENU******

1]FIRST FIT STRATEGY

2]BEST FIT STRATEGY

3]WORST FIT STRATEGY

4]EXIT

Enter your option :
3
600 500 300 200 100
Process of size 212 is allocated to memory block of size 600

Process of size 417 is allocated to memory block of size 500

Process of size 112 is allocated to memory block of size 300

Process of size 426 could not be allocated a memory block

Do you want to continue?(y/n)
y

****MENU******

1]FIRST FIT STRATEGY

2]BEST FIT STRATEGY

3]WORST FIT STRATEGY

4]EXIT

Enter your option :
5

Do you want to continue?(y/n)
y

****MENU******

1]FIRST FIT STRATEGY

2]BEST FIT STRATEGY

3]WORST FIT STRATEGY

4]EXIT

Enter your option :
4

Thank you!

Do you want to continue?(y/n)
n

EXECUTION COMPLETED!!
*/