Greedy Approach

Code for activity selection problem:

```
import java.util.ArrayList;

public class ActivitySelection {
    public static void main(String args[]){
        int start[]={1,3,0,5,8,5};
        int end[]={2,4,6,7,9,9};
        int maxActivity=0;
        ArrayList<Integer> ans= new ArrayList<>();
        // first activity
        maxActivity=1;
        ans.add(0);
        int lastEnd=end[0];

        // if it does not overlap
        for(int i=1;i<end.length;i++){
            if(start[i]>=lastEnd){
                maxActivity++;
                ans.add(i);
                lastEnd=end[i];
            }
        }
        System.out.println("maximum Acivities "+ maxActivity);

        for(int i=0;i<ans.size();i++){
            System.out.println("A"+ans.get(i)+" ");
        }
    }

}

Output:
maximum Acivities 4
A0
A1
A3
A4
```

Activity selection without sorting :

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;

public class ActSelWithoutSorted {
    public static void main(String args[]){
        int start[]={1,3,0,5,8,5};
        int end[]={2,4,6,7,9,9,};
        int maxActivity=0;
        ArrayList<Integer> ans= new ArrayList<>();
        maxActivity=1;

        // if start and end arrays are not sorted
        int activity[][]= new int [start.length][3];
        for(int i=0;i<start.length;i++){
            activity[i][0]=i;
            activity[i][1]=start[i];
            activity[i][2]=end[i];
        }
        //sorting by using comparator
        Arrays.sort(activity,Comparator.comparingDouble(o-> o[2]));
        //creating array list to store ans
        ArrayList<Integer> answer= new ArrayList<>();
        maxActivity=1;
        ans.add(activity[0][0]);
        int lastEnd= activity[0][2];

        for(int i=1;i<end.length;i++){
            if(activity[i][1]>=lastEnd){
            maxActivity++;
            ans.add(activity[i][0]);
            lastEnd=activity[i][2];
            }
        }
        System.out.println("max activity= "+maxActivity);


        }

}
Output:
maximum Acivities 4
A0
```

```
A1
A3
A4
```

Code for Fractional knapsack:

```java
import java.util.*;
public class fractionalKnapsack {
    public static void main(String[] args) {


    int val[]={60,100,120};
    int weight[]={10,20,30};
    int W=50;

    double ratio[][]= new double[val.length][2];
    for(int i=0;i<val.length;i++){
        ratio[i][0]=i;
        ratio[i][1]=val[i]/(double)weight[i];// used double to get exact fraction
    }
    Arrays.sort(ratio,Comparator.comparingDouble(o->o[1]));

    // sorted in ascending order
    // we need to fill the bag based on descending order
    int capacity=W;
    int finalval=0;

    for(int i=val.length-1;i>=0;i--){
        int idx=(int)ratio[i][0];
        if(capacity>=weight[idx]){
            finalval+=val[idx];
            capacity-=weight[idx];
        }else{// if it is in fractional form
            finalval+=ratio[i][1]*capacity;
            capacity=0;
            break;
        }
    }
    System.out.println("max capacity val="+ finalval);



}
}
Output:
max capacity val=240
```

Code for absolute difference:

```
import java.util.*;
public class AbsoluteDifference {
    public static void main(String[] args) {
        int a[]={4,1,8,7};
        int b[]={2,3,6,5};
        Arrays.sort(a);
       Arrays.sort(b);
       int difference=0;
       for(int i=0;i<a.length;i++){
           difference+=Math.abs(a[i]-b[i]);
       }
       System.out.println(difference);
    }
}
Output: 6
```

Maximum number of chain of pairs:

```
import java.util.*;
public class chainofPairsMaxLength {
    public static void main(String[] args) {
        int pairs[][]={{5,24},{39,60},{5,28},{27,40},{50,90}};
        // sorting pairs
        Arrays.sort(pairs,Comparator.comparingDouble(o->o[1]));
        int chainlength=1;
        int lastend=pairs[0][1];
        for(int i=1;i<pairs.length;i++){
            if(pairs[i][0]>lastend){
                chainlength++;
                lastend=pairs[i][1];
            }
        }
        System.out.println("total chains formed "+ chainlength);
    }
}
Output:
total chains formed 3
```

code for Indian coins:

```
import java.util.*;
public class indianCoins {
    public static void main(String[] args) {

    int amount=590;
    //  int coins[]={1,2,5,10,20,50,100,500,2000};

    //  solved using ascending order
    // Arrays.sort(coins);
    // int countcoins=0;
    // for(int i=coins.length-1;i>=0;i--){
    //     if(coins[i]<amount){
    //         while(coins[i]<=amount){
    //             countcoins++;
    //             amount-=coins[i];
    //         }
    //     }
    // }
    // System.out.println("total no of coins is "+countcoins);


    Integer coins[]={1,2,5,10,20,50,100,500,2000};
    // to sort an array in descending order wwe have to use Integer (not int
premitive datatype)
    Arrays.sort(coins,Comparator.reverseOrder());

    ArrayList<Integer> ans= new ArrayList<>();
    int coincount=0;
    for(int i=0;i<coins.length;i++){
        if(coins[i]<=amount){
            while(coins[i]<=amount){
                amount-=coins[i];
                ans.add(coins[i]);
                coincount++;
            }

        }
    }
    System.out.println("coin count = "+ coincount);
    for(int i=0;i<ans.size();i++){
        System.out.print(ans.get(i)+" ");
    }   }

}
Output:
 coin count = 4
500 50 20 20
```

Code for job sequence problem:

```
 import java.util.*;
public class jobsequencingproblem {
    static class job{
        int profit;
        int deadline;
        int id;
        public job(int i, int d, int p){
            id=i;
            deadline=d;
            profit=p;
        }
    }
    public static void main(String[] args) {
        int jobInfo[][]={{4,20},{1,10},{1,40},{1,30}};

        ArrayList<job> jobs= new ArrayList<>(); // object arraylist of job class
       for(int i=0;i<jobInfo.length;i++){
        jobs.add(new job(i, jobInfo[i][0], jobInfo[i][1]));
       }
       Collections.sort(jobs,(obj1,obj2) ->obj2.profit-obj1.profit);

       ArrayList<Integer>sequence=new ArrayList<>();
       int  time=0;

       for(int i=0;i<jobs.size();i++){
        job curr=jobs.get(i);// curr is of job type not int
        if(curr.deadline>time){
            sequence.add(curr.id);
            time++;
        }

       }
       System.out.println("no of jobs= "+ time);
       for(int i=0;i<sequence.size();i++){
        System.out.println(sequence.get(i));
       }
    }
}
Output:
no of jobs= 2
2
0
```

Code for choclate problem:

```
import java.util.*;
public class choclateProblem {
    public static void main(String[] args) {
        int n=4,m=6;
        Integer horizontalcost[]={2,1,3,1,4};
        Integer verticalcost[]={4,1,2};

        Arrays.sort(horizontalcost,Collections.reverseOrder());
        Arrays.sort(verticalcost,Collections.reverseOrder());

        int h=0, v=0;// horizontal and vertical pointers
        int hp=1,vp=1; // horizontal and vertical cuts
        int cost=0;

        while(h<horizontalcost.length && v<verticalcost.length){
            if(verticalcost[v]<=horizontalcost[h]){//horizontal cut
                cost+=(horizontalcost[h]*vp);
                h++;
                hp++;
            }else{//vertical cut
                cost+=(verticalcost[v]*hp);
                vp++;
                v++;
            }
        }
        while(h<horizontalcost.length){
            cost+=(horizontalcost[h]*vp);
            h++;
            hp++;
        }
        while(v<verticalcost.length){
            cost+=(verticalcost[v]*hp);
            vp++;
            v++;
        }
        System.out.println("total cuts "+ cost);

    }
}
Output:
total cuts 42
```