11.1 : Longest palindromic substring

```
public class LongestPalindromicSubstring {
  public String longestPalindrome(String s) {
    if (s == null || s.length() < 1) return "";
    int n = s.length();
    boolean[][] dp = new boolean[n][n];
    int start = 0, maxLength = 1;
    for (int i = 0; i < n; i++) {
      dp[i][i] = true;
    }
    for (int i = 0; i < n - 1; i++) {
      if (s.charAt(i) == s.charAt(i + 1)) {
        dp[i][i + 1] = true;
        start = i;
        maxLength = 2;   } }
    for (int len = 3; len <= n; len++) {
      for (int i = 0; i <= n - len; i++) {
        int j = i + len - 1;
        if (s.charAt(i) == s.charAt(j) && dp[i + 1][j - 1]) {
          dp[i][j] = true;
          start = i;
          maxLength = len; } }
    }
    return s.substring(start, start + maxLength);
  }
  public static void main(String[] args) {
    LongestPalindromicSubstring solution = new LongestPalindromicSubstring();
    System.out.println(solution.longestPalindrome("babad")); // Output: "bab" or "aba"
    System.out.println(solution.longestPalindrome("cbbd")); // Output: "bb"
  }
}
```

11.2 Maximum sub array

Code:

```
public class MaximumSubarray {
    public int maxSubArray(int[] nums) {
        if (nums == null || nums.length == 0) return 0;
        int currentSum = nums[0];
        int maxSum = nums[0];
        for (int i = 1; i < nums.length; i++) {
            currentSum = Math.max(nums[i], currentSum + nums[i]);
            maxSum = Math.max(maxSum, currentSum);
        }
        return maxSum;
    }
    public static void main(String[] args) {
        MaximumSubarray solution = new MaximumSubarray();
        System.out.println(solution.maxSubArray(new int[]{-2,1,-3,4,-1,2,1,-5,4})); // Output: 6
        System.out.println(solution.maxSubArray(new int[]{1})); // Output: 1
        System.out.println(solution.maxSubArray(new int[]{5,4,-1,7,8})); // Output: 23
    }
}
```

11.3 Minimum Cost Tickets

Code:

```
public class MinimumCostForTickets {
   public int mincostTickets(int[] days, int[] costs) {
      boolean[] travelDays = new boolean[366];
      for (int day : days) {
         travelDays[day] = true;
      }
      int[] dp = new int[366];
      for (int i = 1; i <= 365; i++) {
         if (!travelDays[i]) {
            dp[i] = dp[i - 1];
         } else {
            dp[i] = Math.min(
               dp[Math.max(0, i - 1)] + costs[0],
               Math.min(
                  dp[Math.max(0, i - 7)] + costs[1],
                  dp[Math.max(0, i - 30)] + costs[2]));
         }
      }
      return dp[365];
   }
public static void main(String[] args) {
MinimumCostForTickets solution = new MinimumCostForTickets();
System.out.println(solution.mincostTickets(new int[]{1,4,6,7,8,20}, new int[]{2,7,15}));
System.out.println(solution.mincostTickets(new int[]{1,2,3,4,5,6,7,8,9,10,30,31}, new
int[]{2,7,15}));
   }
}
```