問11　次のJavaプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

携帯電話の利用状況に対して，最も安価な料金プラン（割引サービスを含む）を提示するプログラムである。1か月の料金は，基本料金，通話料金及びパケット料金からなる。パケット料金にはオプションの割引サービスがある。

表1に料金プラン，表2にパケット料金の割引サービス，表3にテスト用の利用状況を示す。

なお，パケット料金を計算するとき，パケット数は100パケット単位に切り上げる。

表1　料金プラン

単位　円

プラン名	基本料金	無料通話分(1)	1分当たりの通話料金	100パケット当たりのパケット料金
A	2,000	1,000	40	20
B	3,000	2,000	30	20

注(1)　この料金分までの通話に対する料金は基本料金に含まれている。

表2　パケット料金の割引サービス

サービス名	内容
S1	サービス料金1,000円を支払うと，10,000パケットまでは無料，10,000パケットを超えた分は，100パケット当たり8円とする。

表3　テスト用の利用状況

通話時間（分）	パケット数
30	1,000
30	10,000
300	1,000
300	10,000

(1) クラス`CellPhonePlan`は，料金プランを表す。メソッド`calculateCharge`は，引数で渡された利用状況に対する料金を返す。

(2) クラス CallingPlan は，通話料金を計算するクラスである。メソッド calculateCharge は，引数で渡された通話時間（分単位に切上げ）に対する料金を返す。

(3) インタフェース PacketPlan は，パケット料金を計算するためのインタフェースである。メソッド calculateCharge は，引数で渡されたパケット数に対する料金を返す。

(4) クラス Measured は，従量制のパケット料金（割引サービスなしの場合）を表す。

(5) クラス Tiered は，パケット料金の割引サービス S1 を表す。

(6) クラス CellPhonePlanner は，利用状況に対して，最も安価な料金プラン（割引サービスを含む）を提示する。メソッド getRecommendedPlan は，与えられた利用状況に対して，最も安価な料金プラン（割引サービスを含む）を返す。メソッド main は，テスト用のメインプログラムである。

プログラムの実行結果を図に示す。

```
minutes:   30, packets:   1000 => A      ( 2400)
minutes:   30, packets:  10000 => A S1   ( 3200)
minutes:  300, packets:   1000 => B      (10200)
minutes:  300, packets:  10000 => B S1   (11000)
```

図　プログラムの実行結果

〔プログラム1〕

```
public class CellPhonePlan {
   private CallingPlan callingPlan;
   private PacketPlan packetPlan;

   CellPhonePlan(CallingPlan callingPlan, PacketPlan packetPlan) {
      this.callingPlan = callingPlan;
      this.packetPlan = packetPlan;
   }
   public int calculateCharge(int minutes, int packtes) {
      return callingPlan.calculateCharge(minutes)
         + packetPlan.calculateCharge(packtes);
   }
   public String getName() {
      return callingPlan.getName() + " " + packetPlan.getName();
   }
}
```

〔プログラム2〕

```
public class CallingPlan {
   private String name;
   private int basicCharge;
   private int included;
   private int callingRate;
   CallingPlan(String name, int basicCharge,
               int included, int callingRate) {
      this.name = name;
      this.basicCharge = basicCharge;
      this.included = included;
      this.callingRate = callingRate;
   }
   public String getName() { return name; }
   public int calculateCharge(int minutes) {
      int callingCharge = [ a ];
      if ([ b ])
         callingCharge = 0;
      return basicCharge + callingCharge;
   }
}
```

〔プログラム3〕

```
public interface PacketPlan {
   String getName();
   int calculateCharge(int packets);
}
```

〔プログラム4〕

```
public class Measured [ c ] {
   private int packetRate;

   Measured(int packetRate) {
      this.packetRate = packetRate;
   }
   public String getName() { return ""; }
   public int calculateCharge(int packets) {
      return packetRate * ((packets + 99) / 100);
   }
}
```

〔プログラム5〕

```
public class Tiered [ c ] {
   private String name;
   private int basicCharge;
   private int allowance;
   private int packetRate;

   Tiered(String name, int basicCharge,
          int allowance, int packetRate) {
      this.name = name;
      this.basicCharge = basicCharge;
      this.allowance = allowance;
      this.packetRate = packetRate;
   }
   public String getName() { return name; }
   public int calculateCharge(int packets) {
      int charge = basicCharge;
      if (packets > allowance)
         charge += packetRate * ([ d ]) / 100);
      return charge;
   }
}
```

〔プログラム6〕

```
public class CellPhonePlanner {
   static final CallingPlan[] callingPlans = {
      new CallingPlan("A", 2000, 1000, 40),
      new CallingPlan("B", 3000, 2000, 30) };
   static final PacketPlan[] packetPlans = {
      new Measured(20),
      new Tiered("S1", 1000, 10000, 8) };
   static final CellPhonePlan[] cellPhonePlans = {
      new CellPhonePlan(callingPlans[0], packetPlans[0]),
      new CellPhonePlan(callingPlans[0], packetPlans[1]),
      new CellPhonePlan(callingPlans[1], packetPlans[0]),
      new CellPhonePlan(callingPlans[1], packetPlans[1]) };

   static CellPhonePlan getRecommendedPlan(int minutes,
                                           int packets) {
      int minCharge = [ e ];
      CellPhonePlan recommended = null;
      for (CellPhonePlan cellPhonePlan : cellPhonePlans) {
         int charge = [ f ];
         if (charge <= minCharge) {
            recommended = cellPhonePlan;
            minCharge = charge;
         }
      }
```

```
        return recommended;
    }

    public static void main(String[] args) {
        int[][] testData = {{30, 1000}, {30, 10000},
                            {300, 1000}, {300, 10000} };

        for (int[] data : testData) {
            int minutes = data[0];
            int packets = data[1];
            CellPhonePlan recommended =
                                getRecommendedPlan(minutes, packets);
            System.out.printf("minutes: %4d, packets: %6d => %-6s (%5d)%n",
                              minutes, packets, recommended.getName(),
                              recommended.calculateCharge(minutes,
                                                          packets));
        }
    }
}
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

aに関する解答群

ア　callingRate * minutes

イ　callingRate * minutes - included

ウ　included

エ　included - callingRate * minutes

bに関する解答群

ア　callingCharge < 0　　イ　callingCharge < included

ウ　callingCharge > 0　　エ　callingCharge > included

cに関する解答群

ア　extends CellPhonePlan　　イ　extends PacketPlan

ウ　implements CellPhonePlan　　エ　implements PacketPlan

dに関する解答群

ア　allowance + 99　　イ　packets + 99

ウ　packets + allowance + 99　　エ　packets - allowance + 99

e，fに関する解答群

ア　0

イ　Integer.MIN_VALUE

ウ　Integer.MAX_VALUE

エ　cellPhonePlan.calculateCharge(minutes, packets)

オ　recommended.calculateCharge(minutes, packets)

設問2　パケット通信を大量に行う人を対象として，S1の内容を変更したパケット料金割引サービスS2を追加することにした。表4にサービスの内容，プログラム7にS2を表すクラスを示す。プログラム7中の［　　］に入れる正しい答えを，解答群の中から選べ。

表4　新しく追加するパケット料金の割引サービス

サービス名	内容
S2	サービス料金1,000円を支払うと，10,000パケットまでは無料， 10,000パケットを超えて50,000パケットまでは，100パケット当たり10円， 50,000パケット（合計5,000円）を超えた場合は，一律5,000円とする。

〔プログラム7〕

```
public class TieredBounded [ g ] {
   private int maxCharge;

   TieredBounded(String name, int basicCharge, int allowance,
                 int packetRate, int maxCharge) {
      super(name, basicCharge, allowance, packetRate);
      this.maxCharge = maxCharge;
   }
   public int calculateCharge(int packets) {
      int charge = [ h ];
      if (charge > [ i ])
         charge = maxCharge;
      return charge;
   }
}
```

gに関する解答群

ア extends PacketPlan　　イ extends Tiered

ウ implements PacketPlan　　エ implements Tiered

h, iに関する解答群

ア 0　　イ allowance

ウ basicCharge　　エ basicCharge + maxCharge

オ maxCharge　　カ super.calculateCharge(packets)

キ this.calculateCharge(packets)