Gin:

```
import java.util.*;

public class Gin
{
    String SUITS = "CDHS";
    String RANKS = "A234567891JQK";

     int[] ranks;
     int[] suits;

    public int minDeadWood(String hand)
    {
        ranks = new int[7];
        suits = new int[7];

        StringTokenizer st = new StringTokenizer(hand,SUITS,true);
        int rank=-1, suit=-1;
        int i=0;
        while (st.hasMoreTokens())
        {
            String t = st.nextToken();
            suit = SUITS.indexOf(t);
            if (suit >=0)
            {
                ranks[i] = rank;
                suits[i] = suit;
                i++;
            }
            else
            {
                rank = RANKS.indexOf(t.charAt(0));
            }
        }
        return minDeadWood(ranks,suits);
    }

    int minDeadWood(int[] r, int[] s)
    {
        int N = r.length;

        if (N==0)
        {
            // no cards
            return 0;
```

```
}

 // initialize min with the total value of all cards
int min = 0;
for (int i=0; i<N; i++)
{
   min += val(r[i]);
}

 // since N <= 7 enumerate all subsets (128) looking for safe groups
for (int t=0; t<(1<<N); t++)
{
    // count the number of cards in subset t
   int bitct = 0;
   for (int u=0; u<N; u++)
   {
      if ( (t & (1<<u)) > 0 )
      {
         bitct++;
      }
   }

    // safe groups have size >= 3
   if (bitct>=3)
   {
      // split up the cards into the current subset and the remainder
      int[] rr = new int[bitct];
      int[] ss = new int[bitct];
      int[] rrem = new int[N-bitct];
      int[] srem = new int[N-bitct];
      int b=0;
      int c=0;
      for (int u=0; u<N; u++)
      {
         if ( (t & (1<<u)) > 0 )
         {
            rr[b] = r[u];
            ss[b] = s[u];
            b++;
         }
         else
         {
            rrem[c] = r[u];
            srem[c] = s[u];
            c++;
         }
      }

      if (safe(rr,ss))
```

```
            {
                //  recurse on the remainder
                min = Math.min(min,  minDeadWood(rrem,srem));
            }
        }
    }
    return min;
}

int val(int i)
{
    return Math.min(i+1,10);
}

boolean safe(int[]  r, int[]  s)
{
    int N = r.length;

    if (N<3)
    {
        return false;
    }

     // count the ranks and suits in this set
    boolean[] rs = new boolean[13];
    boolean[] ss = new boolean[4];
    for (int  i=0; i<N; i++)
    {
        rs[ r[i]  ] = true;
        ss[ s[i]  ] = true;
    }
    int rct  = 0;
    for (int  i=0;i<13;i++)
    {
        if (rs[i])  rct++;
    }
    int sct = 0;
    for (int  i=0;i<4;i++)
    {
        if (ss[i])  sct++;
    }

    if (rct  > 1 && sct == 1)
    {
        int rmin = 13;
        int rmax = -1;
        for (int  i=0;i<N;i++)
        {
            rmin = Math.min(rmin,r[i]);
```

```
            rmax = Math.max(rmax,r[i]);
          }
          // same suit, consecutive rank
          return ( rmax+1-rmin == rct );
        }
        else if (sct > 1 && rct == 1)
        {
          // same rank, different suits
          return (sct>=3);
        }
        return false;
    }
}
```