```java
public class Main {
    public static void main(String[] args) {
        // System.out.println(multiples());
        // System.out.println(evenFibonacci());
        // System.out.println(largestPrimeFactor(600851475143L));
        // System.out.println(largestPalindromeProduct());
        // System.out.println(smallestMultiple());
        // System.out.println(sumSquareDifference(100));
        // System.out.println(nthPrime(10001));
        // System.out.println(largestProduct(13));
        // System.out.println(specialPythagoreanTriplet());
        // System.out.println(summationOfPrimes());

    }

    public static int multiples() {
        int sum = 0;
        for (int i = 1; i < 1000; i++) {
            if (i % 3 == 0) {
                sum += i;
            }
            if (i % 5 == 0) {
                sum += i;
            }
            if (i % 3 == 0 && i % 5 == 0) {
                sum -= i;
            }
        }
        return sum;
    }

    public static int evenFibonacci() {
        int sum = 0;
        int a = 1;
        int b = 1;
        int c;
        while (a < 4000000 && b < 4000000) {
            a = a + b;
            c = b;
            b = a + c;
            if (a % 2 == 0) {
                sum += a;
            }
            if (b % 2 == 0) {
                sum += b;
            }
        }
        return sum;
    }

    public static long largestPrimeFactor(long n) {
        for (long i = 2; i < n; i++) {
            if (n % i == 0) {
                return largestPrimeFactor(n/i);
            }
```

```java
    }
    return n;
}

public static int largestPalindromeProduct() {
    int max = 0;
    for (int i = 999; i > 99; i--) {
        for (int j = 999; j > 99; j--) {
            if (isPalindrome(i*j)) {
                max = Math.max(max, i*j);
            }
        }
    }
    return max;
}

public static boolean isPalindrome(int n) {
    String str = String.valueOf(n);
    StringBuilder reversed = new StringBuilder();
    for (int i = 1; i <= str.length(); i++) {
        reversed.append(str.charAt(str.length() - i));
    }
    return str.contentEquals(reversed);
}

public static int smallestMultiple() {
    int lcm = 1;
    while (!divisibleByFirstK(lcm, 20)) {
        lcm++;
    }
    return lcm;
}

public static boolean divisibleByFirstK(int n, int k) {
    for (int i = 1; i < k; i++) {
        if (n % i != 0) {
            return false;
        }
    }
    return true;
}

public static int sumSquareDifference(int n) {
    int difference = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (i != j) {
                difference += i*j;
            }
        }
    }
    return difference;
}

public static int nthPrime(int n) {
```

```
        int i = 0;
        int num = 2;

        while (i != n) {
            if (isPrime(num)) {
                i++;
            }
            num++;
        }

        return num-1;
    }

    public static boolean isPrime(int n) {
        for (int i = 2; i < n; i++) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }

    public static long largestProduct(int n) {
        String str =
"73167176531330624919225119674426574742355349194934969835203127745063262395783180169848
01869478851843858615607891129494954595017379583319528532088055111254069874715852386305
07156932909632952274430435576689664895044524452316173185640309871112172238311362229893
42338030813533627661428280644448664523874930358907296290491560440772390713810515859307
96086670172427121883998797908792274921901699720888093776657273330010533678812202354218
09751254540594752243525849077116705560136048395864467063244157221553975369781797784617
40649551492908625693219784686224828397224137565705605749026140797296865241453510047482
16637048440319989000889524345065854122758866688116427171479924442928230863465674813919
12316282458617866458359124566529476545682848912883142607690042242190226710556263211110
93705442175069416589604080719840385096245544436298123098787992724428490918884580156166
09791913387549920052406368991256071760605886116467109405077541002256983155200055935729
725716362695618826704282524836008232575304207529634 50";
        long max = 0;
        for (int i = 0; i < str.length() - n; i++) {
            long product = 1;
            for (int j = i; j < i + n; j++) {
                product *= Integer.parseInt(str.substring(j, j + 1));
            }
            if(i == 180) {
                int x = 0;
            }
            max = Math.max(max, product);
        }
        return max;
    }

    public static int specialPythagoreanTriplet() {
        for (int a = 1; a < 1000; a++) {
            for (int b = 1; b < 1000; b++) {
                for (int c = 1; c < 1000; c++) {
```

```
                    if (Math.pow(a, 2) + Math.pow(b, 2) == Math.pow(c, 2) && a + b + c
== 1000) {
                        return a*b*c;
                    }
                }
            }
        }
        return -1;
    }

    public static long summationOfPrimes() {
        boolean[] isPrime = new boolean[2000000];
        for (int i = 2; i < 2000000; i++) {
            isPrime[i] = true;
        }

        for (int p = 2; p * p < 2000000; p++) {
            if (isPrime[p]) {
                for (int i = p * p; i < 2000000; i += p) {
                    isPrime[i] = false;
                }
            }
        }

        long sum = 0;
        for (int i = 2; i < 2000000; i++) {
            if (isPrime[i]) {
                sum += i;
            }
        }

        return sum;
    }
}
```

```java
import java.util.ArrayList;
import java.util.Scanner;

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

       Card player = new Card(sc.next());
       Card[] hand = new Card[5];

       for (int i = 0; i < 5; i++) {
           hand[i] = new Card(sc.next());
       }

       ArrayList<Card> sameSuit = new ArrayList<>();
       for (Card card: hand) {
           if (card.suit.equals(player.suit)) {
               sameSuit.add(card);
           }
       }

       Card bestCard;
       if (!sameSuit.isEmpty()) {
           bestCard = sameSuit.get(0);
           for (Card card : sameSuit) {
               if (card.getValue() > bestCard.getValue()) {
                   bestCard = card;
               }
           }
           for (Card card: sameSuit) {
               if (card.getValue() > player.getValue() && card.getValue() <
bestCard.getValue()) {
                   bestCard = card;
               }
           }
           if (bestCard.getValue() < player.getValue()) {
               for (Card card: sameSuit) {
                   if (card.getValue() < bestCard.getValue()) {
                       bestCard = card;
                   }
               }
           }
       } else {
           bestCard = hand[0];
           String order = "CDHS";
           for (Card card: hand) {
               if (card.getValue() < bestCard.getValue()) {
                   bestCard = card;
               } else if (card.getValue() == bestCard.getValue() &&
order.indexOf(card.suit) < order.indexOf(bestCard.suit)) {
                   bestCard = card;
               }
           }
       }
       System.out.println(bestCard);
```

```java
    }

    static class Card {
        String value;
        String suit;
        public Card(String card) {
            value = card.substring(0, 1);
            suit = card.substring(1, 2);
        }

        public int getValue() {
            return switch (value) {
                case "A" -> 1;
                case "T" -> 10;
                case "J" -> 11;
                case "Q" -> 12;
                case "K" -> 13;
                default -> Integer.parseInt(value);
            };
        }

        public String toString() {
            return value + suit;
        }
    }
}
```