Started on	Wednesday, 10 April 2024, 9:51 AM
State	Finished
Completed on	Wednesday, 10 April 2024, 3:18 PM
Time taken	5 hours 26 mins
Grade	**300.00** out of 300.00 (**100**%)

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Momo mempunyai bahasa sendiri untuk berkomunikasi secara rahasia, bahasa tersebut adalah bahasa Umandana. Untuk mengubah kata biasa menjadi kata berbasis Umandana ada beberapa aturan yang harus diterapkan.

1. Huruf a menjadi "aiden"
2. Huruf i menjadi "ipri"
3. Huruf u menjadi "upru"
4. Huruf e menjadi "epre"
5. Huruf o menjadi "opro"
6. Huruf mati yang tidak diikuti huruf vokal menjadi huruf tersebut + "es"
7. Suku kata "ng" yang tidak diikuti huruf vokal menjadi "strengen"
8. Suku kata "ng" yang diikuti huruf vokal tetap menjadi "ng"
9. Suku kata "ny" yang diikuti huruf vokal tetap menjadi "ny"
10. Selain ketentuan di atas, huruf/karakter tidak diubah

Berikut adalah contoh penggunaan bahasa Umandana

Kata awal	Kata dalam Umandana
aku sayang kamu	aidenkupru saidenyaidenstrengen kaidenmupru
kaki kudanil	kaidenkipri kuprudaidennipriles
kristal dingin	kesriprisestaidenles dipringiprines
elang ompong	eprelaidenstrengen opromespoprostrengen
menyanyi sampai monyong	meprenyaidennyipri saidenmespaidenipri mopronyoprostrengen

Sebagai temannya, Momo meminta kamu membuat mesin Umandana Code yang dapat memecahkan pesan yang dibuat dalam bahasa Umandana.

Lengkapilah method pada file Umandana.java.

Submit file **Umandana.java**.

Hint:

- Anda bisa menggunakan method **charAt(int index)** dari kelas String untuk mendapatkan karakter pada indeks tertentu.
- Anda bisa menggunakan method **replaceAll(String regex, String replacement)** untuk mengubah semua substring dengan substring tertentu.

Java 8

Umandana.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	25	Accepted	0.07 sec, 30.02 MB
2	25	Accepted	0.07 sec, 28.92 MB
3	25	Accepted	0.07 sec, 27.85 MB
4	25	Accepted	0.08 sec, 27.80 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Anda diberikan suatu array statik dengan panjang *N* bertipe `Number` ($X_i..X_{N-1}$) dengan tipe data yang mungkin adalah `Byte, Integer, Double, Float, Short, Long`. Anda ditugaskan untuk menghitung total dari setiap elemennya dengan syarat sebagai berikut:

1. Penjumlahan dilakukan dengan sekuensial dari indeks terendah: *i..N-1*
2. Tipe penjumlahan dari setiap elemennya (X_i) bergantung pada elemen tersebut (X_i). Penjumlahan ini akan diawali dengan melakukan casting terhadap setiap elemen yang akan dijumlah. Sebagai contoh, `[Integer(10), Double(20.5), Integer(1)]` akan melalui proses berikut:
   (0) `sum = Integer(10)`
   (1) Karena X_1 bertipe Double, maka dilakukan penjumlahan Double: `sum = Double(sum) + Double(20.5) = Double(30.5)`
   (2) Karena X_2 bertipe Integer, maka dilakukan penjumlahan Integer: `sum = Integer(sum) + Integer(1) = Integer(31)`
3. Jika terdapat *buffer overflow* ketika melakukan penjumlahan (berbeda dengan ketika melakukan casting), hasil dari penjumlahan akan menggunakan nilai maksimum dari tipe penjumlahan jika melewati batas atas dan nilai minimum dari tipe penjumlahan jika melewati batas bawah. Contoh: `Short(32767) + Short(10) = Short(32767)` atau `Short(-32767) + Short(-10) = Short(-32768)`

Contoh:

Masukan	**Keluaran**	**Penjelasan**
[Integer(10), Double(20.5), Integer(1)]	Integer(31)	Rujuk penjelasan di atas
[Short(32767), Short(10)]	Short(32767)	(0) sum = Short(32767) (1) sum = Short(sum) + Short(10) = Short(32767) Pada proses (1), ketika mellakukan penjumlahan, terjadi overflow sehingga menggunakan batas atas tipe penjumlahan, yaitu Short(32767) (Rujuk pada syarat 2).
[Integer(10), Double(20.5), Double(570.5), Byte(011_2), Double(11.5)]	Double(103.5)	(0) sum = Integer(10) (1) sum = Double(sum) + Double(20.5) = Double(30.5) (2) sum = Double(sum) + Double(570.5) = Double(601.0) (3) sum = Byte(sum) + Byte(011_2) = Byte(1011100_2) (4) sum = Double(sum) + Double(11.5) = Double(103.5) Pada proses (3), ketika melakukan casting Byte(sum), terjadi overflow sehingga 601.0 berubah menjadi 89 (601 % 256) (Rujuk pada hint poin 5).

Hint:

- Untuk mengecek apakah suatu kelas merupakan subclass dari kelas lain bisa dengan keyword: `instanceof`
- Untuk mendapatkan nilai maksimum dari subtipe dapat menggunakan `.MAX_VALUE` (eg. `Integer.MAX_VALUE`) dan untuk mendapatkan nilai minimum dari subtipe dapat menggunakan `.MIN_VALUE` (eg. `Integer.MIN_VALUE`)
- Penjumlahan yang menghasilkan *buffer overflow* pada tipe `Long`, `Integer`, `Short`, dan `Byte` akan menghasilkan value yang ter-*wrap*, yaitu positif menjadi negatif dan sebaliknya.
- Penjumlahan yang menghasilkan *buffer overflow* pada tipe `Double`, dan `Float` akan menghasilkan value `Infinity` dan `-Infinity` yang dapat diperiksa dengan static method isInfinite yang dimiliki masing-masing class (`Double.isInfinite(double val)` dan `Float.isInfinite(float val)`).
- Jika terjadi *buffer overflow* ketika melakukan casting, Java akan melakukan modulo terhadap value tersebut.

Lengkapilah metode-metode pada file `NumberSumII.java`. Anda dipersilahkan untuk membuat method tambahan jika diperlukan. Kemudian, Submit file `NumberSumII.java`

Java 8

NumberSumII.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.30 sec, 28.55 MB
2	20	Accepted	0.22 sec, 26.23 MB
3	20	Accepted	0.22 sec, 26.23 MB
4	20	Accepted	0.22 sec, 26.23 MB
5	20	Accepted	0.22 sec, 26.23 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Gabriella adalah calon mahasiswa ITB yang baru saja lolos SNBP. Walau Gabriella tidak masuk sekolah karena libur, ia tetap eksplorasi berbagai hal, salah satunya mengenai kriptografi. Setelah mengetahui bahasa Umandana, Gabriella terinspirasi membuat sebuah algoritma hashing. Algoritma yang dibuat oleh Gabriella melakukan hash dengan cara mengubah ke dalam bahasa Umandana kemudian melakukan shifting huruf sesuai urutan alfabet sebanyak jumlah huruf vokal pada kata awal. Untuk karakter angka, akan dikonversi dahulu menjadi alfabet sesuai urutan pada alfebat dimana 'a' merupakan urutan ke 0. Gabriella memberikan beberapa contoh hasil dari algoritma hash yang ia buat.

Awal	Hashed	Penjelasan
tes	ufqsftft	tes -> tepreses -> ufqsftft, digeser 1 huruf karena hanya terdapat 1 huruf vokal pada tes
tes123	ufqsftftcde	tes123 -> tepreses123; 1 = b; 2 = c; 3 = d; -> ufqsftftcde; 123 = bcd digeser 1 huruf menjadi cde
password123	rckfgpuguuguyqrtqtgufgudef	password123 -> paidensesseswoproresdes123 -> rckfgpuguuguyqrtqtgufgudef, digeser 2 huruf karena ada 'a' dan 'o' pada password

Untuk membuktikan algoritma hash yang Gabriella buat dapat bekerja, ia ingin membuat sebuah sistem manajemen akun. Bantulah Gabriella untuk membuat sistem tersebut.

Pelajari dan lengkapi kelas **Account**, **Ucrypt** dan **AccountManager**. Gunakanlah **`Umandana.java`** yang telah Anda buat untuk membantu implementasi.

`Account.java`
Kelas ini merepresentasikan akun pada sistem. Anda perlu melengkapi constructor dan methods yang terdapat pada kelas Account.

`Ucrypt.java`
Kelas ini merepresentasikan algoritma hashing.
Mohon lengkapi method berikut:

- hash(String word)
- compare(String plain, String hashed)

`AccountManager.java`
Kelas ini merepresentasikan sistem manajemen akun. AccountManager mempunyai list dari Account yang telah didaftarkan. Untuk menambahkan akun ada beberapa validasi yang harus dipenuhi. Anda perlu melengkapi constructor dan methods yang terdapat pada kelas AccountManager.

Kumpulkan file **Account.java Ucrypt.java AccountManager.java** dalam satu file zip!

Ekstrak file - file yang dibutuhkan pada zip berikut.
Anda hanya diperbolehkan mengimpor java.util.Arrays.

Untuk memudahkan debugging berikut adalah test case yang diperiksa

Test Case	Kelas/method yang diperiksa
#1	Account
#2	Ucrypt
#3	AccountManager.addAccount
#4	AccountManager.login

Note:

- Terdapat newline disetiap output.
- Anda diperbolehkan menambahkan method tambahan tanpa mengubah method yang ada.

Hint:

- Manfaatkan method dari class Character untuk mengerjakan, seperti isLetterOrDigit(char ch), isLetter(char ch), isDigit(char ch).
- Untuk melakukan shifting character Anda bisa memanfaatkan type char di Java yang disimpan dalam 16-bit Unicode sehingga Anda hanya perlu type casting char dengan mengeser 1 karakter. Contohnya: x = ((char) ('a' + 1)), maka x akan bernilai 'b'.

Java 8

Praktikum.zip

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	25	Accepted	0.07 sec, 28.92 MB
2	25	Accepted	0.08 sec, 28.40 MB
3	25	Accepted	0.20 sec, 33.93 MB
4	25	Accepted	0.20 sec, 33.79 MB

◄ Praktikum 4

Jump to...