Tugas Praktikum PBO Kelas D

**Assertion & Exception**

1. Assertion

Saki adalah orang yang gemar membaca buku. Karena semua koleksi buku miliknya telah habis ia baca dan ia juga tidak ingin meminjam buku dari perpustakaan karena malas keluar kamar dan berinteraksi dengan orang, ia pun memutuskan untuk membeli buku pada perpustakaan digital. Karena ia adalah pengguna baru, ia diharuskan untuk melakukan registrasi terlebih dahulu pada aplikasi tersebut. Pada aplikasi tersebut memiliki field username, email, dan age. Dimana persyaratan untuk username adalah minimal terdiri dari 5 karakter, tidak lebih dari 20 karakter serta hanya boleh berisi huruf, angka, dan underscore ( \_ ) serta harus diawali dengan huruf. Persyaratan untuk email adalah harus berbentuk email yang valid. Persyaratan untuk usia adalah diantara 17 tahun hingga 99 tahun. Buatlah sebuah validasi input pada aplikasi tersebut.

* 1. UserInputValidator class

|  |
| --- |
| public class UserInputValidator {  public static void validateUsername(String username) {  // TODO: buat assertion yang memenuhi persyaratan username  }  public static void validateEmail(String email) {  // TODO: buat assertion yang memenuhi persyaratan email  }  public static void validateAge(int age) {  // TODO: buat assertion yang memenuhi persyaratan usia  }  } |

* 1. UserRegistration class

|  |
| --- |
| import java.util.Scanner;  public class UserRegistration {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter username: ");  String username = scanner.nextLine();  UserInputValidator.validateUsername(username);  System.out.print("Enter email address: ");  String email = scanner.nextLine();  UserInputValidator.validateEmail(email);  System.out.print("Enter age: ");  int age = scanner.nextInt();  UserInputValidator.validateAge(age);  System.out.println("User registration successful!");  }  } |

Jalankan program dengan command

|  |
| --- |
| javac UserInputValidator.java  javac UserRegistration.java  java -enableassertions UserRegistration |

Hint:

Untuk validasi username dan email bisa pakai regex

Spill tipis:

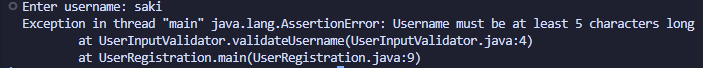
Buat yang suka tantangan bisa tetap menutup mata

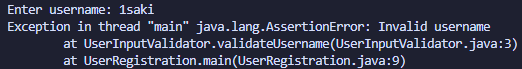
But here you then, (silahkan blok bar hitam dan hilangkan highlight untuk melihat isi)

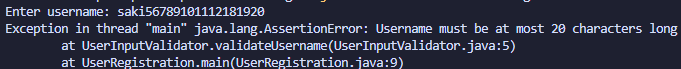
regex for username validation : [a-zA-Z][a-zA-Z0-9\_]+

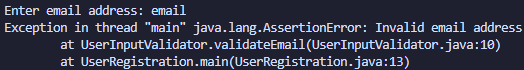
regex for email validation : [a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}

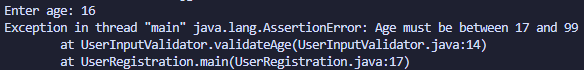
Contoh hasil testing

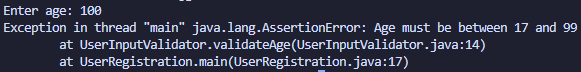


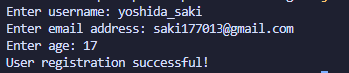












1. Exception

Setelah puas melihat-lihat koleksi buku dan memilih buku yang ingin dibeli, Saki pun melakukan checkout dan melakukan pembayaran via mobile banking. Pada aplikasi mbanking tersebut terdapat akun nasabah yang memiliki data berapa jumlah saldo yang dimiliki oleh nasabah tersebut. Jika pada saat ingin melakukan pembayaran dan jumlah saldo yang dimiliki tidak mencukupi, maka pembayaran tersebut gagal. Buatlah kondisi exception yang memenuhi kondisi tersebut.

* 1. InsufficientFundsException class

|  |
| --- |
| public class InsufficientFundsException extends Exception {  public InsufficientFundsException(String message) {  super(message);  }  } |

* 1. BankAccount class

|  |
| --- |
| public class BankAccount {  private double balance;  private final double MIN\_BALANCE = 100.0; // Saldo minimal  public BankAccount(double initialBalance) {  this.balance = initialBalance;  }  public void deposit(double amount) {  balance += amount;  System.out.println("Deposited: $" + amount);  }  public void withdraw(double amount) {  /\* TODO: buatlah logic yang memenuhi persyaratan diatas  dengan menggunakan exception \*/  }  public double getBalance() {  return balance;  }  } |

* 1. BankApp class

|  |
| --- |
| public class BankApp {  public static void main(String[] args) {  /\* TODO: buatlah main yang akan melakukan operasi transaksi  dan akan menampilkan pesan bila terjadi kesalahan  skenario bebas berkreasi, tidak harus sama seperti contoh dibawah  BankAccount account = new BankAccount(500.0);  // skenario saldo mencukupi  account.withdraw(200.0);  System.out.println("Saldo saat ini: $" + account.getBalance());  // skenario saldo tidak mencukupi  account.withdraw(400.0);  \*/  }  } |

Jalankan program dengan command

|  |
| --- |
| javac InsufficientFundsException.java  javac BankAccount.java  javac BankApp.java  java BankApp |

Hint:

try, catch, throw, throws

Contoh hasil testing:

Saldo awal 500 > withdraw 200 > withdraw 400

