Nama : Kelvianto Pratama Harum

NIM : 200210500016

MK : Pemrograman Lanjut

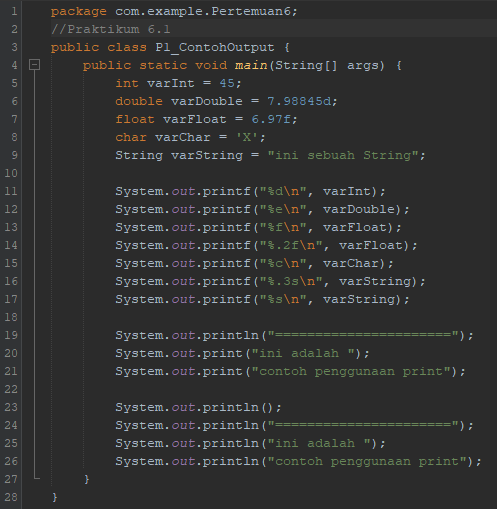
Dosen : Muhammad Fajar B, S.Pd., M.Cs.

**PERTEMUAN VI**

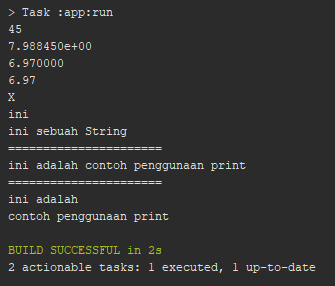
**FILE AND EXCEPTION HANDLING**

1. **Praktikum 6.1 – ContohOutput.java**

Source code:



Output:



Penjelasan:

Pada praktikum ini dibuat sebuah class dengan nama **P1\_ContohOutput** dengan sebuah **main** method.

Pada bagian **main** method, dideklarasikan sejumlah variabel, yaitu:

* **varInt** dengan tipe data **integer** dan nilai **45**
* **varDouble** dengan tipe data **double** dan nilai **7.98845d**
* **varFloat** dengan tipe data **float** dan nilai **6.97f**
* **varChar** dengan tipe data **char** dan nilai **‘X’**
* **varString** dengan tipe data **String** dan nilai **“ini sebuah String”**

Kemudian variabel tersebut dioutput ke console dengan **printf** yaitu method yang menampilkan data sesuai dengan format string yang diberikan:

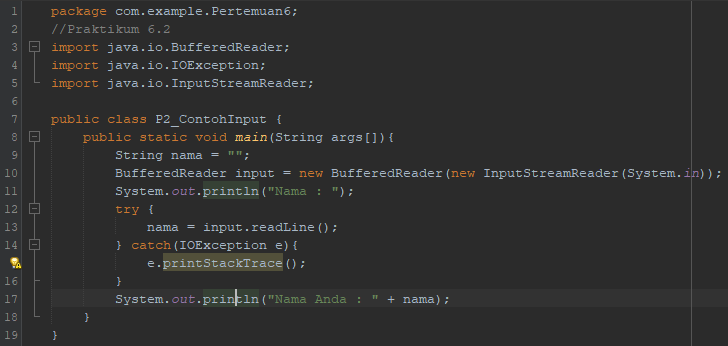
* Variabel **varInt** dioutput dengan format **‘%d’** yaitu format yang digunakan untuk menampilkan data **integer (decimal, base-10)**, sehingga data yang ditampilkan pada console adalah **45** seperti pada output.
* Variabel **varDouble** dioutput dengan format **‘%e’** yaitu format yang digunakan untuk menampilkan data **float atau double dengan notasi scientific**, sehingga data yang ditampilkan pada console adalah **7.988450e+00** (ini sama seperti **7.988450 x 100)** seperti pada output.
* Variabel **varFloat** dioutput dengan format **‘%f’** yaitu format yang digunakan untuk menampilkan data **float**, sehingga data yang ditampilkan pada console adalah **6.970000** seperti pada output.
* Variabel **varFloat** dioutput dengan format **‘%.2f’** yaitu format yang digunakan untuk menampilkan data **float dengan 2 angka setelah koma**, sehingga data yang ditampilkan pada console adalah **6.97** seperti pada output.
* Variabel **varChar** dioutput dengan format **‘%c’** yaitu format yang digunakan untuk menampilkan data **char (atau character)**, sehingga data yang ditampilkan pada console adalah **‘X’** seperti pada output.
* Variabel **varString** dioutput dengan format **‘%.3s’** yaitu format yang digunakan untuk menampilkan data **String sebanyak 3 karakter dari awal**, sehingga data yang ditampilkan pada console adalah **ini** seperti pada output.
* Variabel **varString** dioutput dengan format **‘%s’** yaitu format yang digunakan untuk menampilkan data **String**, sehingga data yang ditampilkan pada console adalah **“ini sebuah String”** seperti pada output.

Kemudian dioutput string berikut:

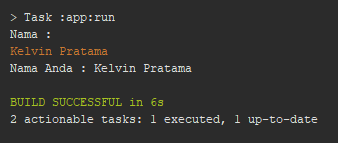
* **“===================”** menggunakan method **println** yang artinya String yang diberikan akan diberikan **newline (\n) atau baris baru** pada akhir String.
* **“ini adalah “** menggunakan method **print** yang artinya String akan dioutput apa adanya tanpa modifikasi
* **“contoh penggunaan print”** menggunakan method **print** yang artinya String akan dioutput apa adanya tanpa modifikasi
* Sebuah baris kosong menggunakan **println**
* **“===================”** menggunakan method **println** yang artinya String yang diberikan akan diberikan **newline (\n) atau baris baru** pada akhir String.
* **“ini adalah “** menggunakan method **println** yang artinya String yang diberikan akan diberikan **newline (\n) atau baris baru** pada akhir String.
* **“contoh penggunaan print”** menggunakan method **println** yang artinya String yang diberikan akan diberikan **newline (\n) atau baris baru** pada akhir String.

1. **Praktikum 6.2 – ContohInput.java**

Source code:



Output:



Penjelasan:

Pada praktikum ini dilakukan import **method** bawaan java seperti:

* **java.io.BufferedReader** dan **java.io.InputStreamReader** yang digunakan untuk membaca input keyboard user melalui console.
* **java.io.IOException** yang digunakan untuk menghandle exception terkait **input dan output** pada program.

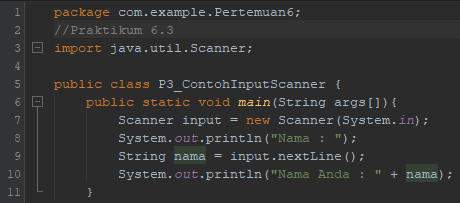
Kemudian dibuat sebuah class dengan nama **P2\_ContohInput** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

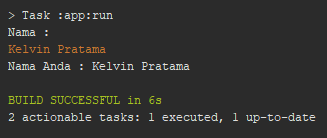
* Variabel **nama** dideklarasikan dengan tipe data **String** dan diinisialisasi dengan string kosong
* Kemudian dibuat sebuah object **input** dari class **BufferedReader** dan **InputStreamReader** yang digunakan sebagai interface antara input keyboard user dan program.
* Output sebuah string **“Nama: ”** dengan method **println** yang artinya string akan diikuti dengan baris baru.
* Lalu, program akan mencoba mengambil **input** dari user lalu dimasukkan dalam variabel **nama**
* Jika program mendapatkan eksepsi **IOException**, eksepsi tersebut ditangkap ke dalam object **e**, lalu hasil tracing stack callnya ditampilkan dalam console.
* Lalu, program akan menampilkan string **“Nama Anda: ”** diikuti dengan nilai variabel **nama**

1. **Praktikum 6.3 – ContohInputScanner.java**

Source code:



Output:



Penjelasan:

Pada praktikum ini dilakukan import **method** bawaan java seperti:

* **java.util.Scanner** yang digunakan untuk membaca input keyboard user melalui console.

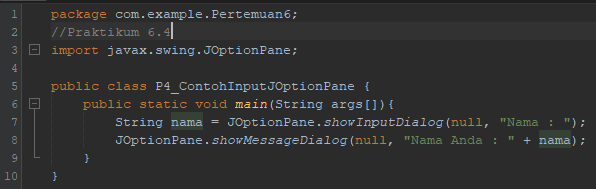
Kemudian dibuat sebuah class dengan nama **P3\_ContohInputScanner** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

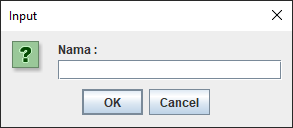
* Dibuat sebuah object interface bernama **input** yang merupakan object dari class **java.util.Scanner**. Object ini digunakan untuk membaca input keyboard user melalui console.
* Ditampilkan string **“Nama: ”** dengan method **println** yang artinya string diikuti dengan baris baru
* Lalu, program akan membaca input user pada object **input** menggunakan method **nextLine()** lalu nilainya disimpan dalam variabel **nama**
* Lalu, program akan menampilkan string **“Nama Anda: ”** diikuti dengan nilai variabel **nama**

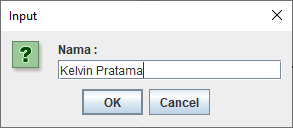
1. **Praktikum 6.4 – ContohInputJOptionPane.java**

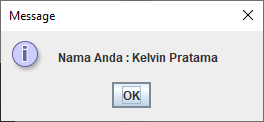
Source code:



Output:







Penjelasan:

Pada praktikum ini dilakukan import **method** bawaan java seperti:

* **javax.swing.JOptionPane** yang digunakan untuk komponen GUI bawaan Java

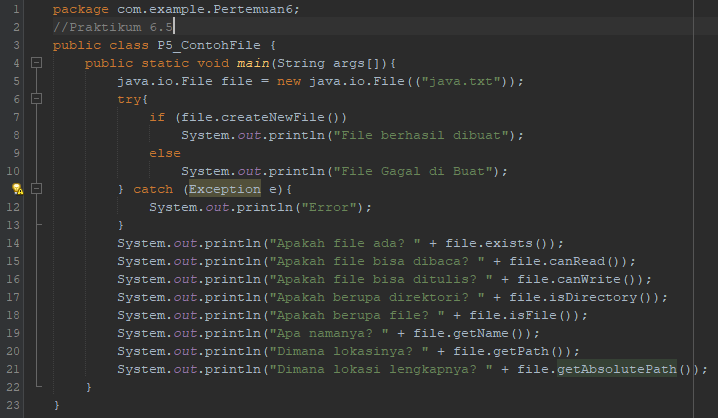
Kemudian dibuat sebuah class dengan nama **P4\_ContohInputJOptionPane** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

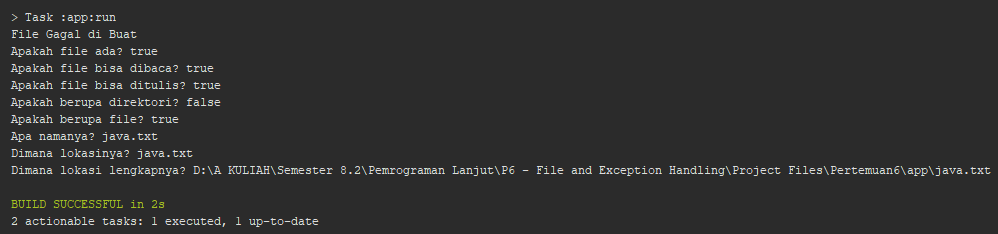
* Sebuah panel GUI Input Dialog dari **JOptionPane** dibuat yang digunakan untuk menampilkan sebuah kolom textbox dengan placeholder **“Nama: “**
* Input dari Panel tersebut disimpan dalam sebuah variabel dengan nama **‘nama’** dan tipe data **String**.
* Kemudian, sebuah panel message dialog ditampilkan berisikan string **“Nama Anda: ”** diikuti dengan isi variabel **nama**.

1. **Praktikum 6.5 – ContohFile.java**

Source code:



Output:



Penjelasan:

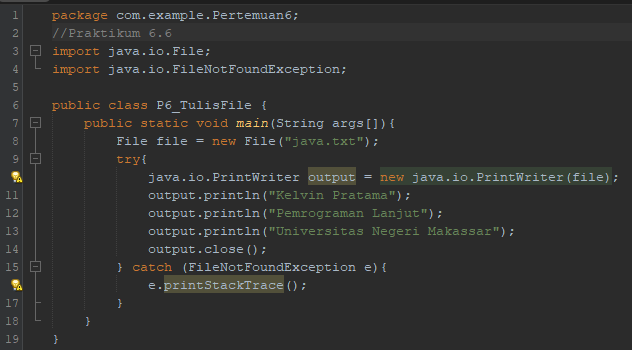
Pada praktikum ini, dibuat sebuah class dengan nama **P5\_ContohFile** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

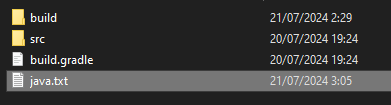
* Dibuat sebuah object dengan nama **file** yang merupakan object dari **java.io.File** untuk file **java.txt**
* Kemudian, program akan mencoba membuat file dengan nama tersebut menggunakan method **createNewFile()**. Jika berhasil, maka akan ditampilkan **“File berhasil dibuat”** pada console, jika tidak maka akan ditampilkan **“File Gagal di Buat”** pada console.
* Dan jika percobaan tersebut gagal (menghasilkan eksepsi **Exception**) maka akan ditangkap dalam object **e**. Lalu, program akan menampilkan **“Error”** pada console
* Lalu, program akan mengecek apakah file tersebut ada atau tidak menggunakan method **exists()** yang kemudian hasilnya ditampilkan bersama dengan string **“Apakah file ada? ”**
* Lalu, program akan mengecek apakah file tersebut bisa dibaca menggunakan method **canRead()** yang kemudian hasilnya ditampilkan bersama dengan string **“Apakah file bisa dibaca? ”**
* Lalu, program akan mengecek apakah file tersebut bisa ditulis menggunakan method **canWrite()** yang kemudian hasilnya ditampilkan bersama dengan string **“Apakah file bisa ditulis? ”**
* Lalu, program akan mengecek apakah file tersebut merupakan direktori menggunakan method **isDirectory()** yang kemudian hasilnya ditampilkan bersama dengan string **“Apakah berupa direktori? ”**
* Lalu, program akan mengecek apakah file tersebut merupakan sebuah file menggunakan method **isFile()** yang kemudian hasilnya ditampilkan bersama dengan string **“Apakah berupa file? ”**
* Lalu, program akan mengecek nama file tersebut menggunakan method **getName()** yang kemudian hasilnya ditampilkan bersama dengan string **“Apa namanya? ”**
* Lalu, program akan mengecek lokasi relatif file tersebut menggunakan method **getPath()** yang kemudian hasilnya ditampilkan bersama dengan string **“Dimana lokasinya? ”**
* Lalu, program akan mengecek lokasi absolut (lengkap) file tersebut menggunakan method **getAbsolutePath()** yang kemudian hasilnya ditampilkan bersama dengan string **“Dimana lokasi lengkapnya? ”**

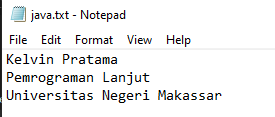
1. **Praktikum 6.6 – TulisFile.java**

Source code:



Output:





Penjelasan:

Pada praktikum ini dilakukan import **method** bawaan java seperti:

* **java.io.File** yang digunakan untuk mengakses sebuah file.
* **java.ioFileNotFoundException** yang digunakan untuk eksepsi jika file tidak ditemukan dari class **java.io.File**

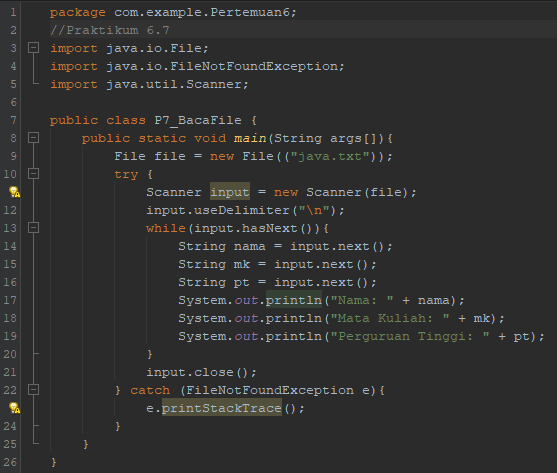
Kemudian dibuat sebuah class dengan nama **P6\_TulisFile** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

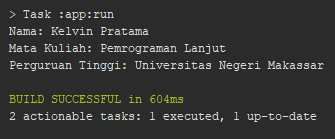
* Dibuat sebuah object **file** untuk mengakses file **java.txt** menggunakan class **java.io.File**
* Kemudian, program mencoba untuk membuat sebuah interface writer **output** pada object **file** menggunakan class **java.io.PrintWriter**.
* Lalu, program mencoba menulis **“Kelvin Pratama”** pada object **output**
* Lalu, program mencoba menulis **“Pemrograman Lanjut”** pada object **output**
* Lalu, program mencoba menulis **“Universitas Negeri Makassar”** pada object **output**
* Lalu, program menutup file tersebut untuk dibaca menggunakan method **close()** pada object **output**.
* Jika program terjadi eksepsi **FileNotFoundException (file tidak ditemukan)** akan ditangkap pada object **e**. Lalu ditampilkan call stack terakhirnya hingga terjadi eksepsi menggunakan method **printStackTrace()**

1. **Praktikum 6.7 – BacaFile.java**

Source code:



Output:



Penjelasan:

Pada praktikum ini dilakukan import **method** bawaan java seperti:

* **java.io.File** yang digunakan untuk mengakses sebuah file.
* **java.ioFileNotFoundException** yang digunakan untuk eksepsi jika file tidak ditemukan dari class **java.io.File**
* **java.util.Scanner** yang digunakan untuk membaca input dari sebuah interaksi eksternal, contohnya file atau user input.

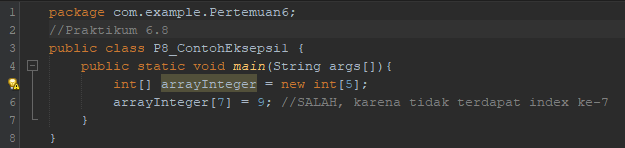
Kemudian dibuat sebuah class dengan nama **P7\_TulisFile** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

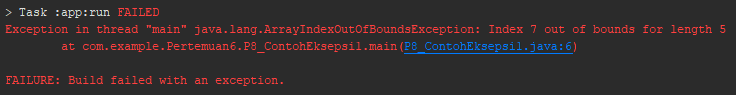
* Dibuat sebuah object **file** untuk mengakses file **java.txt** menggunakan class **java.io.File**
* Kemudian, program akan mencoba membuat sebuah object **input** untuk membaca file dari object **file** menggunakan class **java.util.Scanner**.
* Lalu, didefinisikan pemisah antar data dengan pemisah **“\n”** atau baris baru.
* Lalu, perulangan dengan kondisi **selama object input masih memiliki data belum dibaca**.
* Kemudian, program akan membaca object **input** untuk data berikutnya (**next()**) yang kemudian disimpan dalam variabel **nama** dengan tipe data **String**.
* Lalu, program akan membaca object **input** untuk data berikutnya (**next()**) yang kemudian disimpan dalam variabel **mk** dengan tipe data **String**.
* Lalu, program akan membaca object **input** untuk data berikutnya (**next()**) yang kemudian disimpan dalam variabel **pt** dengan tipe data **String**.
* Lalu, program akan menampilkan string **“Nama: ”** dengan isi variabel **nama**
* Lalu, program akan menampilkan string **“Mata Kuliah: ”** dengan isi variabel **mk**
* Lalu, program akan menampilkan string **“Perguruan Tinggi: ”** dengan isi variabel **pt**
* Lalu, program akan mengakhir pembacaan file dengan method **close** pada object **input**.
* Jika program terjadi eksepsi **FileNotFoundException (file tidak ditemukan)** akan ditangkap pada object **e**. Lalu ditampilkan call stack terakhirnya hingga terjadi eksepsi menggunakan method **printStackTrace()**

1. **Praktikum 6.8 – ContohEksepsi1.java**

Source code:



Output:



Penjelasan:

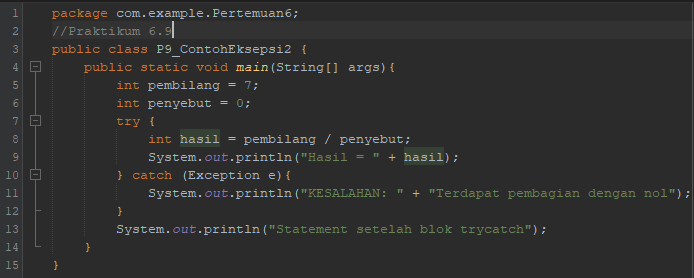
Pada praktikum ini, dibuat sebuah class dengan nama **P8\_ContohEksepsi1** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

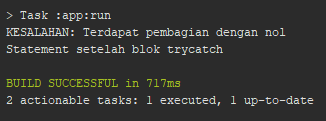
* Dibuat sebuah array tipe data **integer** dengan nama **arrayInteger** dengan member **{1,2,3,4,5}**
* Kemudian, program akan mencoba mengakses array dengan indeks **7** (data ke-8, zero-based index)untuk diisi dengan nilai **9**. Eksekusi ini akan gagal dan akan membangkitkan eksepsi **ArrayIndexOutOfBoundsException** seperti yang terlihat pada output console.

1. **Praktikum 6.9 – ContohEksepsi2.java**

Source code:



Output:



Penjelasan:

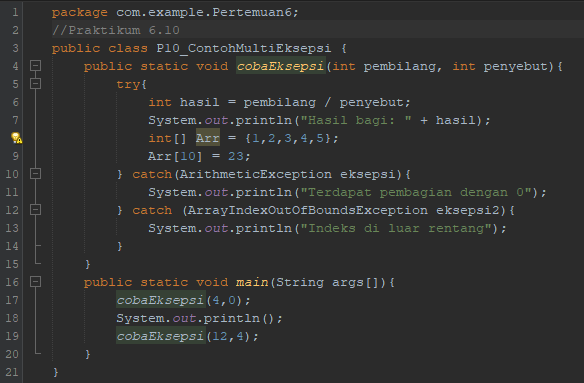
Pada praktikum ini, dibuat sebuah class dengan nama **P9\_ContohEksepsi2** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

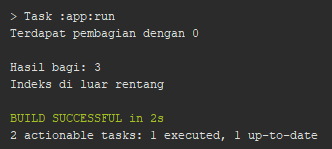
* Dideklarasikan variabel **pembilang** dengan tipe data **integer** dan diinisialisasi dengan nilai **7**.
* Dideklarasikan variabel **penyebut** dengan tipe data **integer** dan diinisialisasi dengan nilai **0**.
* Kemudian, program mencoba untuk menghitung nilai variabel **pembilang** dibagi dengan variabel **penyebut** lalu hasilnya disimpan dalam variabel **hasil** dengan tipe data integer
* Lalu, program akan menampilkan string **“Hasil = ”** dengan nilai variabel **hasil**
* Jika program membangkitkan eksepsi **Exception** maka akan ditangkap pada object **e**, lalu akan ditampilkan string **“KESALAHAN: Terdapat pembagian dengan nol”** pada console
* Lalu setelah itu program akan menampilkan string **“Statement setelah blok try-catch”**

1. **Praktikum 6.10 – ContohMultiEksepsi.java**

Source code:



Output:



Penjelasan:

Pada praktikum ini, dibuat sebuah class dengan nama **P9\_ContohEksepsi2** dengan sebuah method **main** dan method **cobaEksepsi**.

Pada method **cobaEksepsi** diperlukan 2 parameter untuk pemanggilannya, yaitu variabel **pembilan** dan **penyebut** dengan tipe data **integer**. Dalam method ini dilakukan eksekusi sebagai berikut:

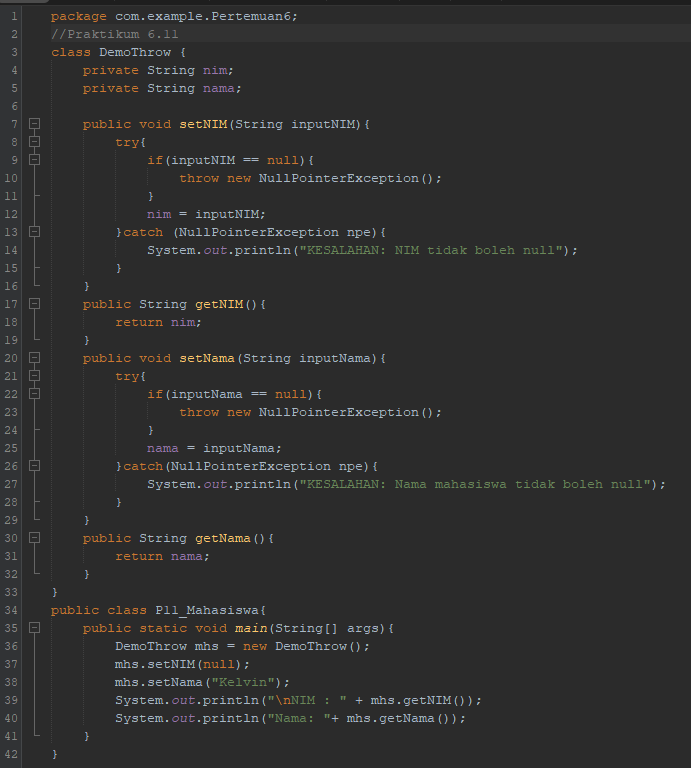
* Program mencoba melakukan eksekusi menggunakan **try-catch**
* Dalam blok **try**, program menghitung nilai dari operasi matematika nilai variabel **pembilang** dibagi dengan **penyebut** lalu hasilnya disimpan dalam variabel **hasil** dengan tipe data **integer**. Jika terdapat operasi yang tidak valid, seperti pembagian dengan nilai **0**, program akan membangkitkan eksepsi **ArithmeticException**.
* Lalu, program akan menampilkan string **“Hasil bagi: ”** dengan nilai variabel **hasil**.
* Kemudian, program membuat array dari tipe data **integer** dengan nama **Arr** dan member **{1,2,3,4,5}**
* Kemudian, program mencoba mengakses indeks ke-**10** (**data ke-11, zero-based index) array Arr** untuk diisi dengan nilai **23**. Eksekusi ini akan membangkitkan eksepsi **ArrayIndexOutOfBoundsException**, karena tidak ada index ke-10 pada array tersebut.
* Jika program membangkitkan eksepsi **ArithmeticException** maka akan ditangkap pada object **e**. Lalu, akan ditampilkan string **“Terdapat pembagian dengan 0”** pada console
* Jika program membangkitkan eksepsi **ArrayIndexOutOfBoundsException** maka akan ditangkap pada object **e**. Lalu, akan ditampilkan string **“Indeks diluar rentang”** pada console

Pada **main** method, dilakukan eksekusi sebagai berikut:

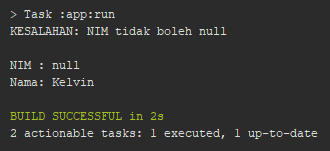
* Pemanggilan method **cobaEksepsi** dengan argumen nilai **4** dan **0**.
* Pemanggilan method **cobaEksepsi** dengan argumen nilai **12** dan **4**.

1. **Praktikum 6.11 – Mahasiswa.java**

Source code:



Output:



Penjelasan:

Pada praktikum ini, dideklarasikan 2 class yaitu **DemoThrow** dan **Mahasiswa**.

Class **DemoThrow** dideklarasikan dengan modifier **default**.

Pada class **DemoThrow**, dideklarasikan 2 fielddengan modifier **private** dengan tipe data **String**, yaitu **nim** dan **nama**.

Kemudian, dideklarasikan method setter untuk field **nim** dengan nama **setNIM** dengan parameter pemanggilan variabel **inputNIM**. Eksekusi didalam method ini adalah sebagai berikut:

* Program mencoba untuk memeriksa jika **inputNIM** adalah **null**, jika iya maka program akan membangkitkan eksepsi **NullPointerException**.
* Program memasukkan nilai **inputNIM** dalam field **nim**.
* Jika program membangkitkan eksepsi **NullPointerException** maka akan ditangkap kedalam object **e**. Lalu akan ditampilkan **“KESALAHAN: NIM tidak boleh null”** pada console.

Lalu, dideklarasikan method getter untuk field **nim** dengan nama **getNIM**. Method ini akan mengembalikan nilai field **nim** pada pemanggil method.

Kemudian, dideklarasikan method setter untuk field **nama** dengan nama **setNama** dengan parameter pemanggilan variabel **inputNama**. Eksekusi didalam method ini adalah sebagai berikut:

* Program mencoba untuk memeriksa jika **inputNama** adalah **null**, jika iya maka program akan membangkitkan eksepsi **NullPointerException**.
* Program memasukkan nilai **inputNama** dalam field **nama**.
* Jika program membangkitkan eksepsi **NullPointerException** maka akan ditangkap kedalam object **e**. Lalu akan ditampilkan **“KESALAHAN: Nama Mahasiswa tidak boleh null”** pada console.
* Lalu, dideklarasikan method getter untuk field **nama** dengan nama **getNama**. Method ini akan mengembalikan nilai field **nama** pada pemanggil method.

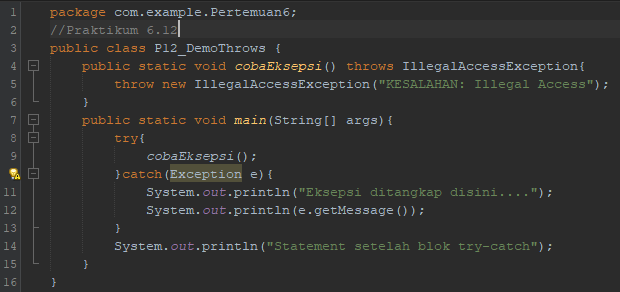
Kemudian, dibuat sebuah class dengan nama **P11\_Mahasiswa** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

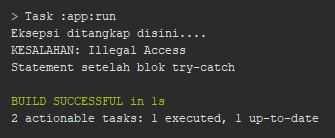
* Dibuat sebuah object **mhs** dari class **DemoThrow**
* Dipanggil method **setNIM** dengan argumen **null** pada object **mhs**
* Dipanggil method **setNama** dengan argumen **“Kelvin”** pada object **mhs**
* Kemudian, program menampilkan string **“NIM: ”** dengan nilai balik pemanggilan method **getNIM()** pada object **mhs**
* Kemudian, program menampilkan string **“Nama: ”** dengan nilai balik pemanggilan method **getNama()** pada object **mhs**

1. **Prakikum 6.12 – DemoThrows.java**

Source code:



Output:



Penjelasan:

Pada praktikum ini, dibuat sebuah class dengan nama **P12\_DemoThrows** dengan sebuah method **main** dan method **cobaEksepsi**.

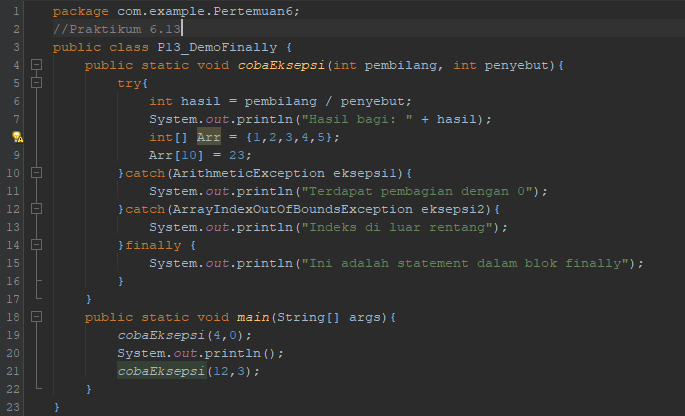
Pada method **cobaEksepsi**, akan dibangkitkan eksepsi **IllegalAccessException** dengan pesan **“KESALAHAN: Illegal Access”**

Pada **main** method, dilakukan eksekusi sebagai berikut:

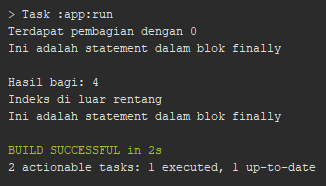
* Program mencoba melakukan eksekusi method **cobaEksepsi()**
* Jika program membangkitkan **Exception** maka akan ditangkap pada object **e**. Lalu program akan menampilkan string **“Eksepsi ditangkap disini…”** dan pesan dari eksepsi tersebut.
* Lalu program akan menampilkan string **“Statement setelah blok try-catch”** setelah blok **try-catch**

1. **Praktikum 6.13 – DemoFinally.java**

Source code:



Output:



Penjelasan:

Pada praktikum ini, dibuat sebuah class dengan nama **P13\_DemoFinally** dengan sebuah method **main** dan method **cobaEksepsi**.

Pada method **cobaEksepsi** diperlukan 2 parameter untuk pemanggilannya, yaitu variabel **pembilang** dan **penyebut** dengan tipe data **integer**. Dalam method ini dilakukan eksekusi sebagai berikut:

* Program mencoba melakukan eksekusi menggunakan **try-catch**
* Dalam blok **try**, program menghitung nilai dari operasi matematika nilai variabel **pembilang** dibagi dengan **penyebut** lalu hasilnya disimpan dalam variabel **hasil** dengan tipe data **integer**. Jika terdapat operasi yang tidak valid, seperti pembagian dengan nilai **0**, program akan membangkitkan eksepsi **ArithmeticException**.
* Lalu, program akan menampilkan string **“Hasil bagi: ”** dengan nilai variabel **hasil**.
* Kemudian, program membuat array dari tipe data **integer** dengan nama **Arr** dan member **{1,2,3,4,5}**
* Kemudian, program mencoba mengakses indeks ke-**10** (**data ke-11, zero-based index) array Arr** untuk diisi dengan nilai **23**. Eksekusi ini akan membangkitkan eksepsi **ArrayIndexOutOfBoundsException**, karena tidak ada index ke-10 pada array tersebut.
* Jika program membangkitkan eksepsi **ArithmeticException** maka akan ditangkap pada object **e**. Lalu, akan ditampilkan string **“Terdapat pembagian dengan 0”** pada console
* Jika program membangkitkan eksepsi **ArrayIndexOutOfBoundsException** maka akan ditangkap pada object **e**. Lalu, akan ditampilkan string **“Indeks diluar rentang”** pada console
* Lalu, program akan menjalankan perintah pada blok **finally** terlepas dari adanya eksepsi yang dibangkitkan atau tidak. Program akan menampilkan string **“Ini adalah statement dalam blok finally”**

Pada **main** method, dilakukan eksekusi sebagai berikut:

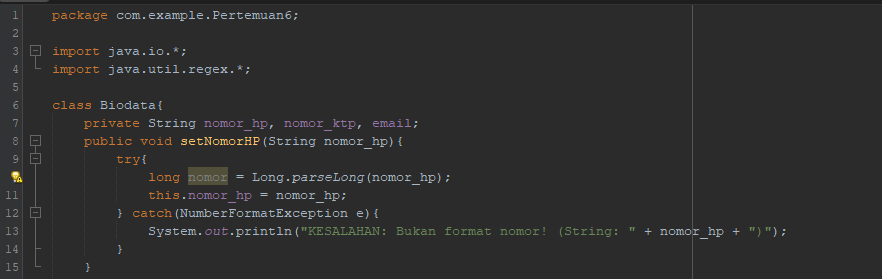
* Pemanggilan method **cobaEksepsi** dengan argumen nilai **4** dan **0**.
* Pemanggilan method **cobaEksepsi** dengan argumen nilai **12** dan **4**.

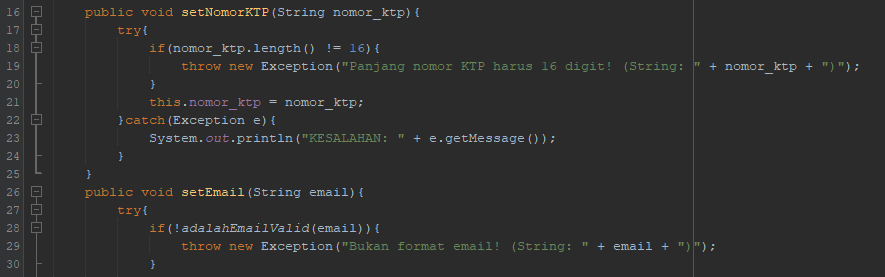
1. **Tantangan**

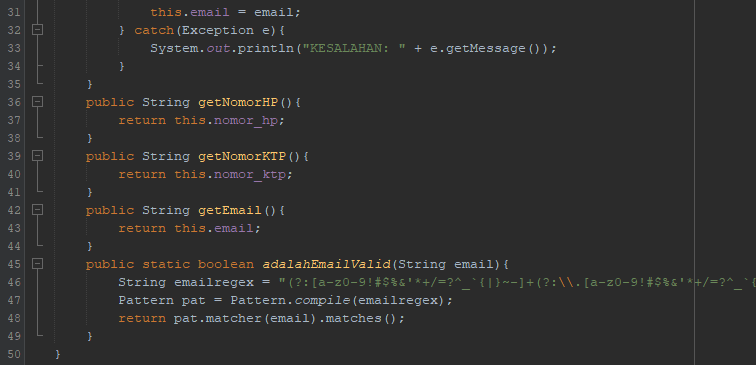
Buatlah sebuah program penanganan eksepsi untuk inputan data yang tidak sesuai dengan format penulisan seperti berikut:

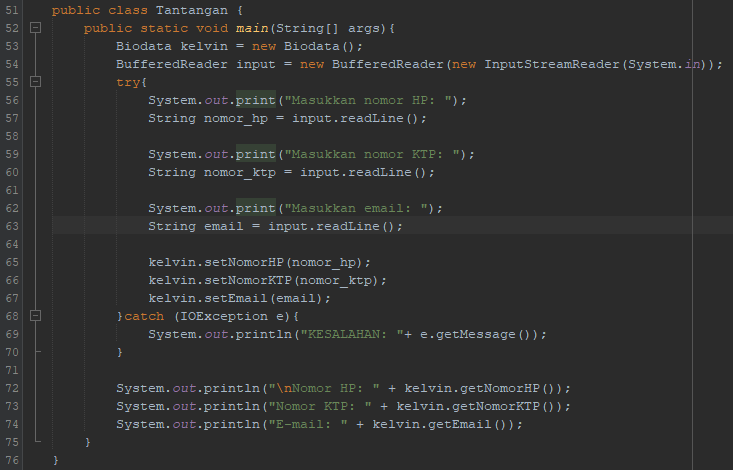
1. Nomor handphone : harus berupa nomor
2. Nomor KTP : harus berjumlah 16 digit angka
3. Alamat email : harus sesuai dengan format penulisan email yang baku

Source code:

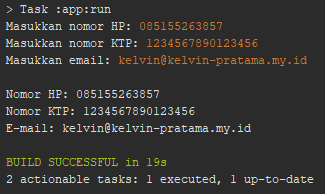


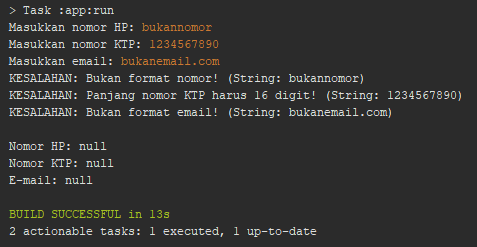






Output:





Penjelasan:

Pada praktikum ini dilakukan import **method** bawaan java seperti:

* **java.io.\*** yang digunakan untuk mengimport seluruh class dalam **java.io**.
* **java.util.regex.\*** yang digunakan untuk mengimport seluruh class dalam **java.util.regex**

Kemudian dibuat sebuah class dengan modifier **default** dengan nama **Biodata**.

Pada class **Biodata**, dilakukan eksekusi sebagai berikut:

* Dideklarasikan **field** dengan modifier **private** dan tipe data **String** dengan nama **nomor\_hp, nomor\_ktp,** dan **email**.
* Kemudian dideklarasikan method setter untuk field **nomor\_hp** dengan nama **setNomorHP** dan parameter pemanggilan variabel **nomor\_hp** bertipe data **String**.
* Dalam method ini, program akan mencoba mengkonversi nilai **nomor\_hp** ke tipe data **Long** dan nilainya disimpan pada dummy variabel **nomor**.
* Lalu, program akan menyimpan nilai argumen **nomor\_hp** pada **field nomor\_hp**.
* Jika program membangkitkan eksepsi **NumberFormatException**, akan ditampilkan **“KESALAHAN: Bukan format nomor! (String: ”** diikuti nilai argumen **nomor\_hp**.
* Kemudian dideklarasikan method setter untuk field **nomor\_ktp** dengan nama **setNomorKTP** dan parameter pemanggilan variabel **nomor\_ktp** bertipe data **String**.
* Dalam method ini, program akan memeriksa jika panjang argumen **nomor\_ktp** tidak sama dengan **16**. Jika iya, maka program akan membangkitkan eksepsi **Exception** dengan pesan **“Panjang nomor KTP harus 16 digit! (String: ”** diikuti dengan nilai argumen **nomor\_ktp**
* Lalu, program akan menyimpan nilai argumen **nomor\_ktp** pada **field nomor\_ktp**.
* Jika program membangkitkan eksepsi **Exception**, akan ditampilkan **“KESALAHAN: ”** diikuti pesan eksepsi.
* Kemudian dideklarasikan method setter untuk field **email** dengan nama **setEmail** dan parameter pemanggilan variabel **email** bertipe data **String**.
* Dalam method ini, program akan memeriksa nilai balik pemanggilan method **adalahEmailValid** dengan argumen nilai variabel **email** adalah **false**. Jika iya, maka program akan membangkitkan eksepsi **Exception** dengan pesan **“Bukan format email (String: ”** diikuti dengan nilai argumen **nomor\_ktp**
* Lalu, program akan menyimpan nilai argumen **email** pada **field email**.
* Jika program membangkitkan eksepsi **Exception**, akan ditampilkan **“KESALAHAN: ”** diikuti pesan eksepsi.
* Kemudian, dideklarasikan method getter untuk field **nomor\_hp** dengan nama **getNomorHP()**
* Kemudian, dideklarasikan method getter untuk field **nomor\_ktp** dengan nama **getNomorKTP()**
* Kemudian, dideklarasikan method getter untuk field **email** dengan nama **getEmail()**
* Kemudian, dideklarasikan method **adalahEmailValid** dengan tipe data balik **boolean** dan parameter pemanggilan **email** dengan tipe data **String**
* Dalam method ini, dilakukan eksekusi pengecekan jika string yang diberikan dalam pemanggilan sesuai dengan format **regular expression** email yang dideklarasikan pada string **emailregex**. Jika iya, program akan mengembalikan nilai **true**. Jika tidak, maka program akan mengembalikan nilai **false**.

Kemudian dibuat sebuah class dengan nama **Tantangan** dengan sebuah method **main**.

Pada **main** method, dilakukan eksekusi sebagai berikut:

* Dibuat sebuah object dengan nama **kelvin** dari class **Biodata**.
* Dibuat sebuah object interface **input** untuk membaca user-input dari console.
* Kemudian, program akan mencoba menjalankan blok **try-catch**
* Dalam blok **try**, program menampilkan string **“Masukkan nomor HP: ”**
* Lalu, program akan mengambil nilai input dari console lalu memasukkannya pada variabel **nomor\_hp** dengan tipe data **String**.
* Lalu, program menampilkan string **“Masukkan nomor KTP: ”**
* Lalu, program akan mengambil nilai input dari console lalu memasukkannya pada variabel **nomor\_ktp** dengan tipe data **String**.
* Lalu, program menampilkan string **“Masukkan email: ”**
* Lalu, program akan mengambil nilai input dari console lalu memasukkannya pada variabel **email** dengan tipe data **String**.
* Lalu, dipanggil method **setNomorHP** dengan argumen variabel **nomor\_hp** pada object **kelvin**.
* Lalu, dipanggil method **setNomorKTP** dengan argumen variabel **nomor\_ktp** pada object **kelvin**.
* Lalu, dipanggil method **setEmail** dengan argumen variabel **email** pada object **kelvin**.
* Jika program membangkitkan eksepsi **IOException** pada blok **try**, maka blok **catch** akan dieksekusi. Program akan menampilkan string **“KESALAHAN: ”** dengan pesan eksepsi yang diberikan.
* Lalu, program akan menampilkan **“Nomor HP: ”** diikuti dengan nilai balik pemanggilan method **getNomorHP()** pada object **kelvin**
* Lalu, program akan menampilkan **“Nomor KTP: ”** diikuti dengan nilai balik pemanggilan method **getNomorKTP()** pada object **kelvin**
* Lalu, program akan menampilkan **“E-mail: ”** diikuti dengan nilai balik pemanggilan method **getEmail()** pada object **kelvin**