MAKALAH, PERCOBAAN, LATIHAN, DAN TUGAS

MODUL PRAKTIKUM 9

Disusun sebagai salah satu tugas

mata kuliah PBO I



Patricia Joanne

140810160065

Dikumpulkan tanggal

7 November 2017

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2017

**Tugas Pendahuluan**

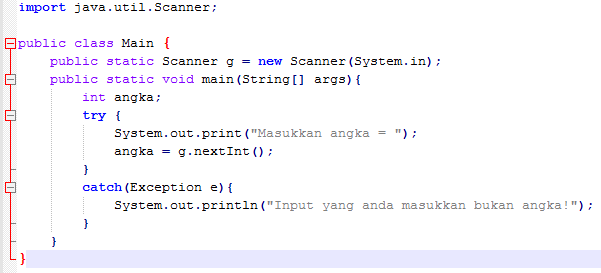
Buatlah review 1 halaman mengenai penanganan exception dengan cara melempar exception dan berikan 1 contoh program.

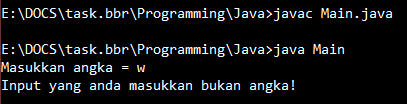
Exception adalah suatu kondisi abnormal yang terjadi pada saat menjalankan program karena dalam Java segala sesuatu merupakan objek, maka exception juga direpresentasikan dalam sebuah objek yang menjelaskan tentang exception tersebut.

Terdapat dua penanganan exception yaitu menangani sendiri exception tersebut atau meneruskannya ke luar dengan cara membuat objek tentang exception tersebut dan melemparkannya (throw) keluar agar ditangani oleh kode yang memanggil method (method yang di dalamnya terdapat exception) tersebut.

Ada lima keyword yang digunakan oleh Java untuk menangani exception yaitu try, catch, finally, throw dan throws.

Contoh program sederhana:

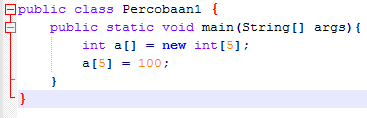


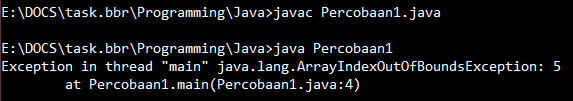


**Praktikum 9**

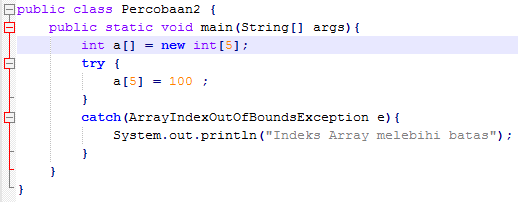
**Exception Class**

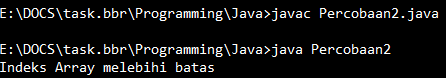
Percobaan 1: Jalankan program di bawah ini Bagaimana output program? Jelaskan!



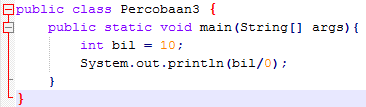


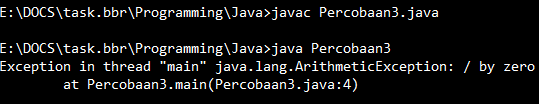
Percobaan 2: Memahami cara menangkap Exception dengan tipe ArrayIndexOutOfBounds-Exception

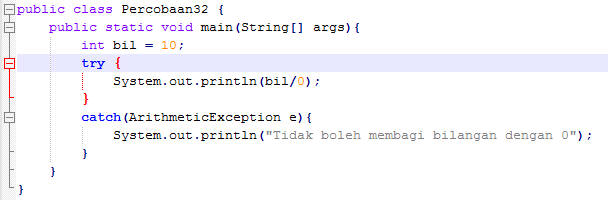


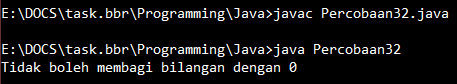


Percobaan 3: Jalankan percobaan 3, bagaimana output program? Perbaiki dengan Percobaan32 untuk menangani exception.

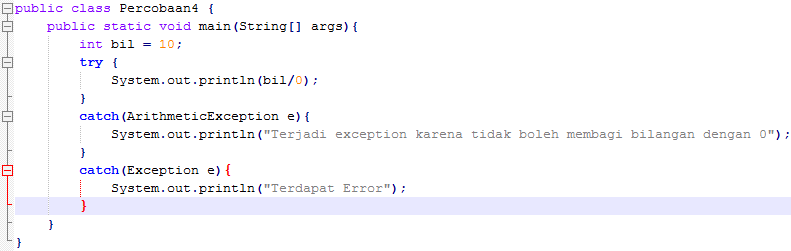


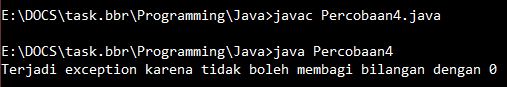




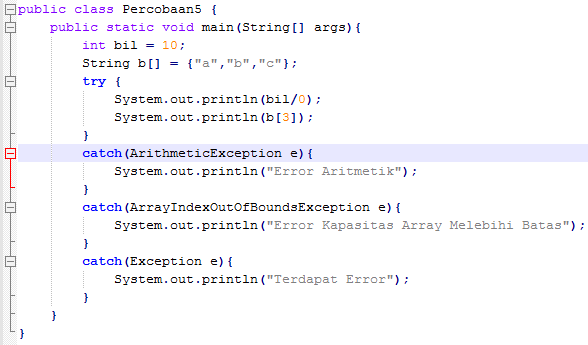


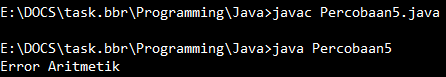
Percobaan 4: Memahami try bertingkat

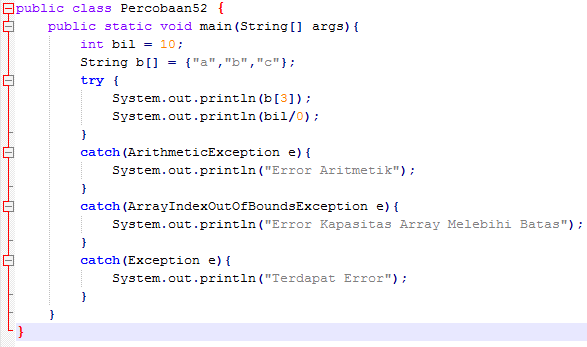


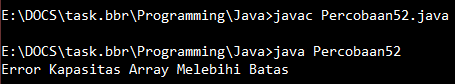


Percobaan 5: Bandingkan output dua program di bawah ini! Jelaskan!

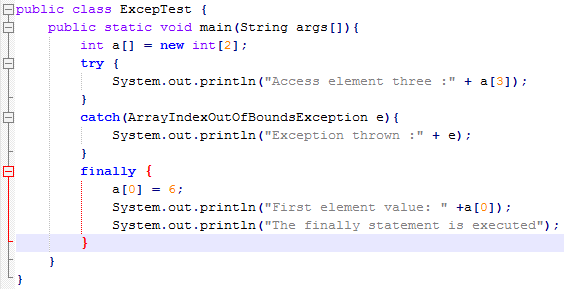


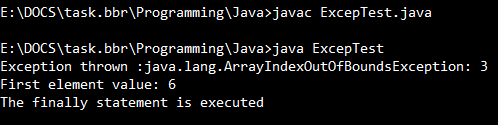




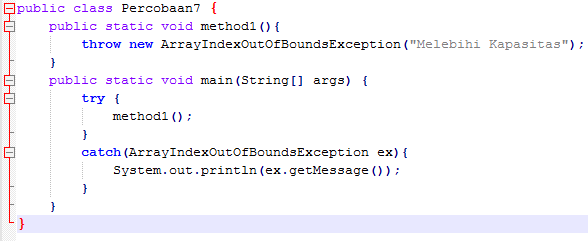


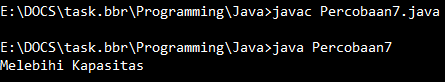
Percobaan 6: Penggunaan finally



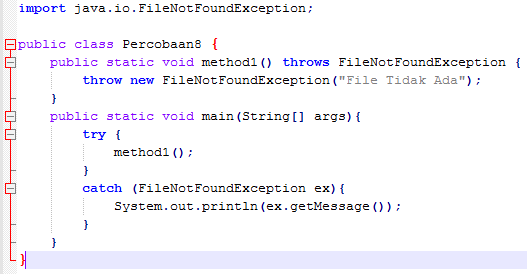


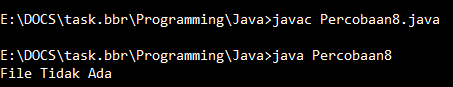
Percobaan 7: Method yang melempar unchecked exception



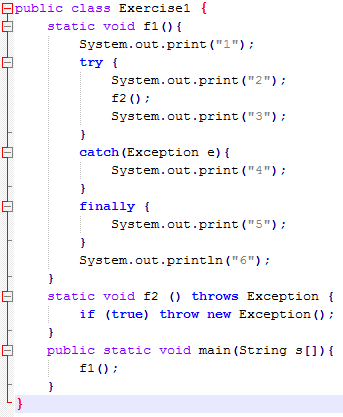


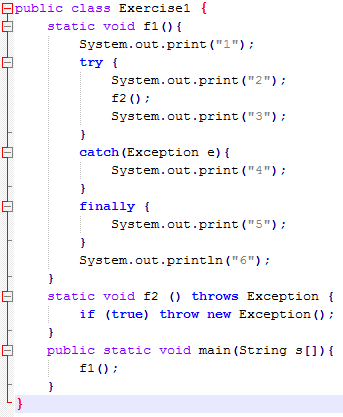
Percobaan 8: Method yang melempar checked exception

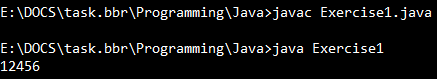




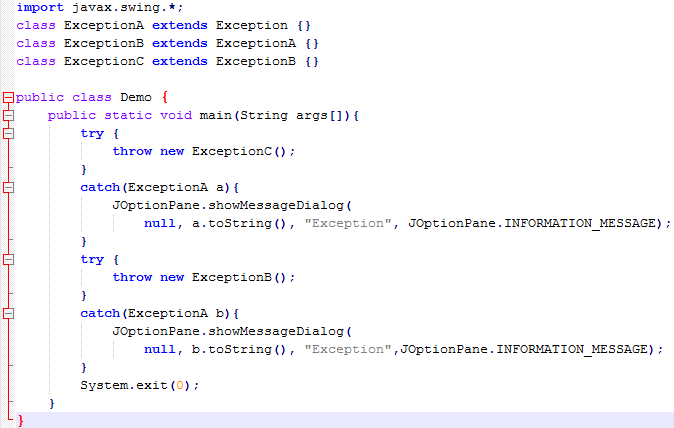
Percobaan 9: Memahami mengenai mekanisme exception



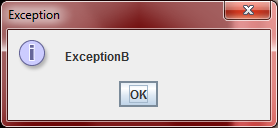
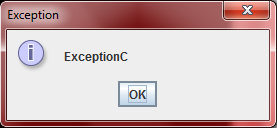




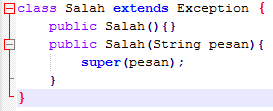
Percobaan 10: Menggunakan konsep Inheritance untuk membuat superclass dan subclass exception. Program menangani exception dengan menangkap subclass exception dengan superclass.

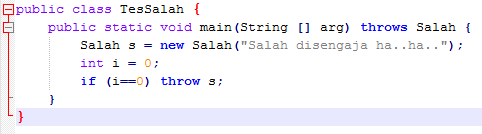


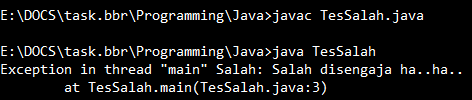




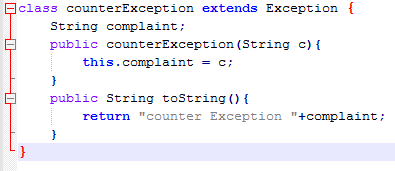
Percobaan 11: Membuat exception sendiri

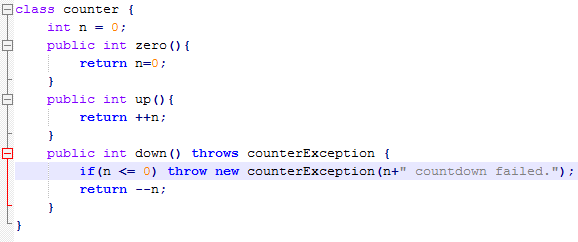


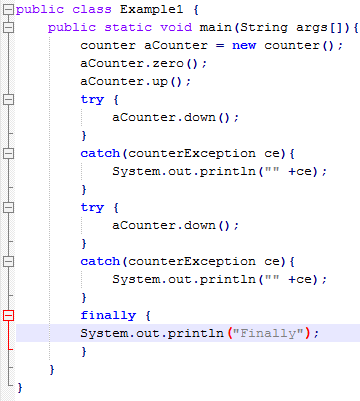


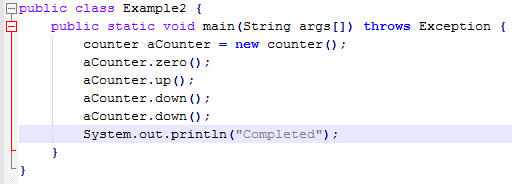


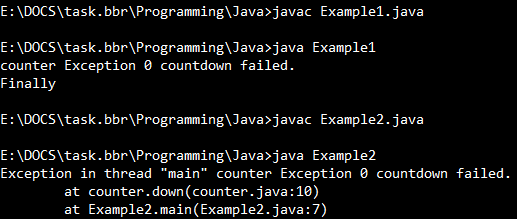
Percobaan 12: Membuat exception sendiri







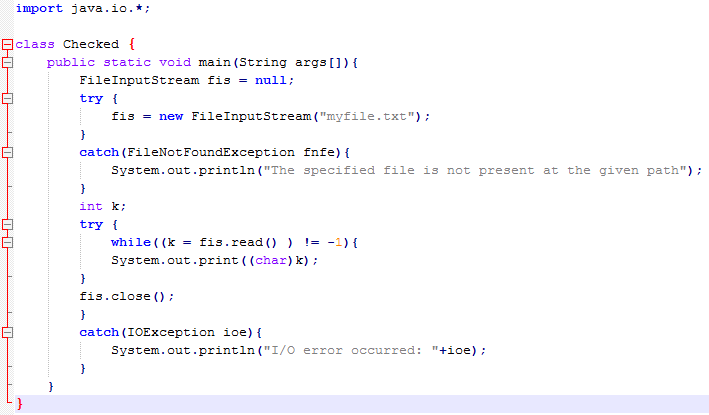


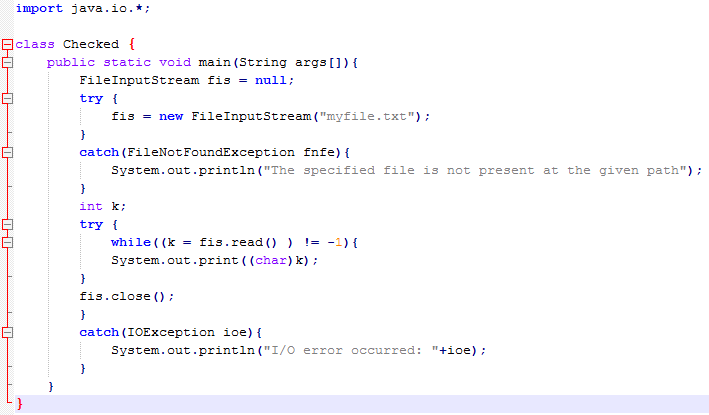


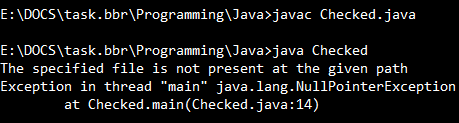
Latihan:

1. Semua exception yang berasal dari java.lang.RuntimeException adalah unchecked exceptions, sedangkan exception lainnya yang tidak berasal dari java.lang.RuntimeException adalah checked exceptions. Jelaskan mengenai unchecked exceptions dan checked exceptions, berikan contoh!
2. Checked Exception adalah eksepsi yang diantisipasi oleh programmer untuk dihandle dalam program dan terjadi dikarenakan oleh kondisi luar yang siap muncul saat program berjalan. Misalnya membuka file yang tidak ada atau gangguan jaringan. Yang termasuk checked exception adalah class java.lang.Throwable dan semua subclassnya, kecuali class dan subclass dari java.lang.Error dan java.lang.RuntimeError.

Contoh:

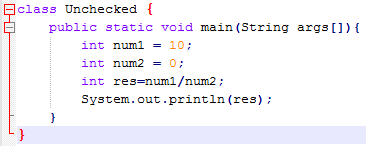


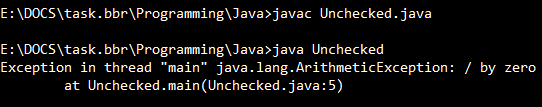




1. Unchecked Exception bisa muncul dari kondisi yang merepresentasikan adanya bug atau situasi yang secara umum dianggap terlalu sulit bagi program untuk menghandlenya. Disebut sebagai unchecked karena kita tidak perlu mengeceknya atau melakukan sesuatu jika kondisi ini terjadi. Eksepsi yang muncul dari kategori situasi yang merepresentasikan bug ini disebut sebagai runtime exception. Misalnya mengakses array melebihi size yang dimilikinya. Yang termasuk unchecked exception adalah java.lang.Error dan subclassnya serta java.lang.RuntimeException dan subclassnya.

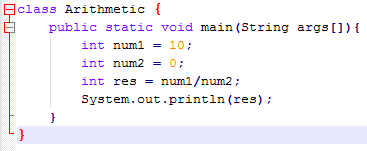
Contoh:

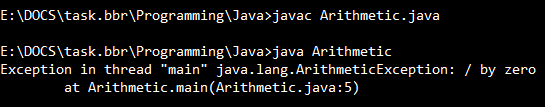




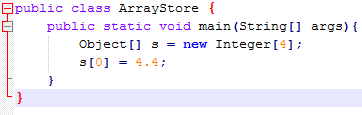
1. Buatlah contoh program untuk menangani exception dengan cara menangkap exception seperti di bawah ini:

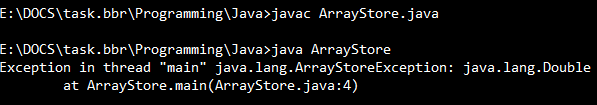
* ArithmeticException



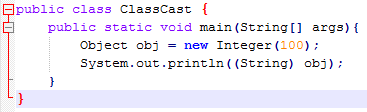


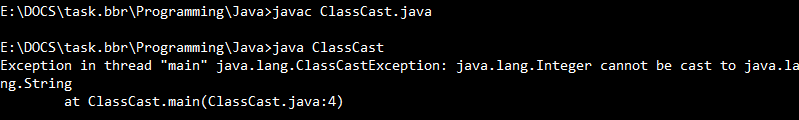
* ArrayStoreException



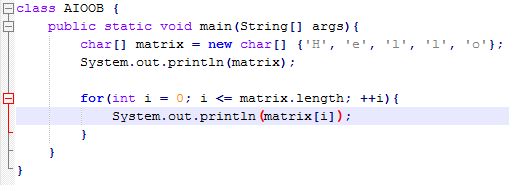


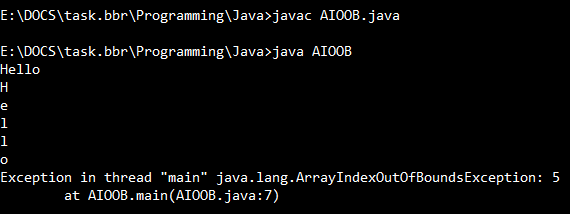
* ClassCastException



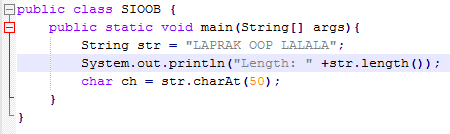


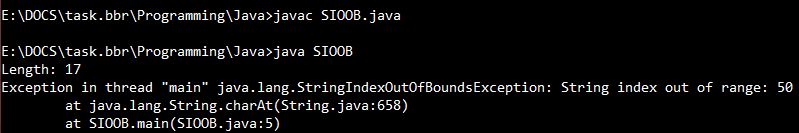
* ArrayIndexOutOfBoundsException



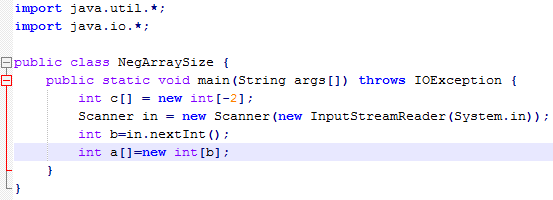


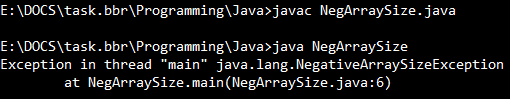
* StringIndexOutOfBoundsException



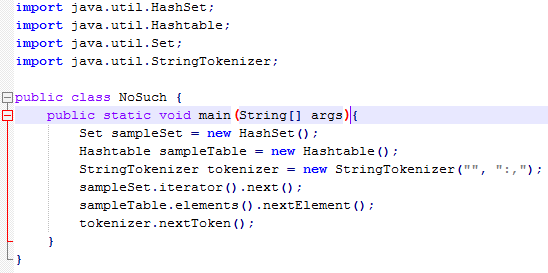


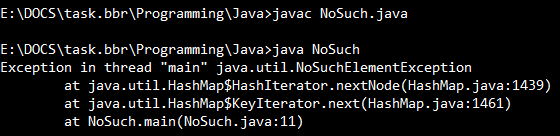
* NegativeArraySizeException



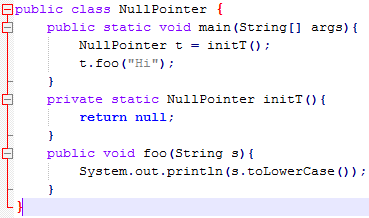


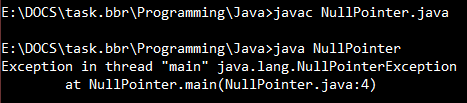
* NoSuchElementException



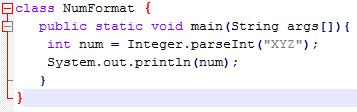


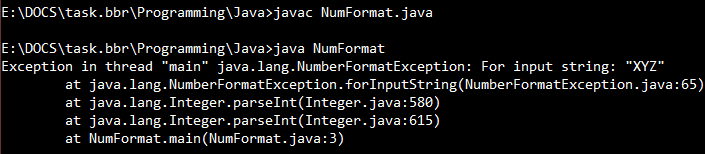
* NullPointerException



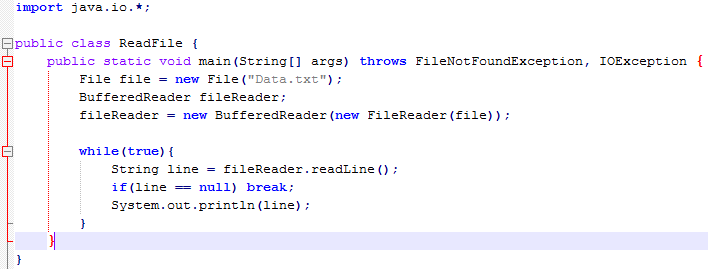


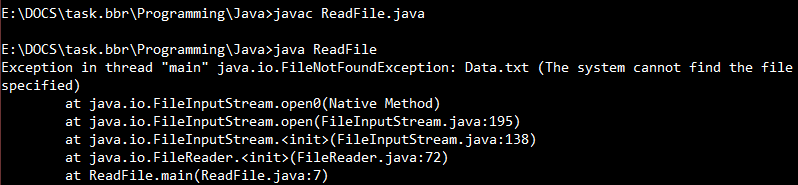
* NumberFormatException





1. Terdapat dua cara untuk menangani Exception yaitu dengan menangkap Exception dan melempar Exception. Lakukan penanganan exception dengan **melempar Exception menggunakan throw**. Berilah penjelasan (apakah program termasuk unchecked exceptions atau checked exceptions)!

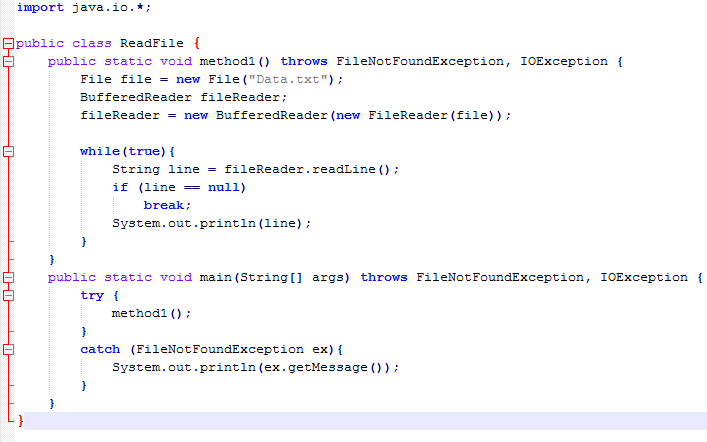


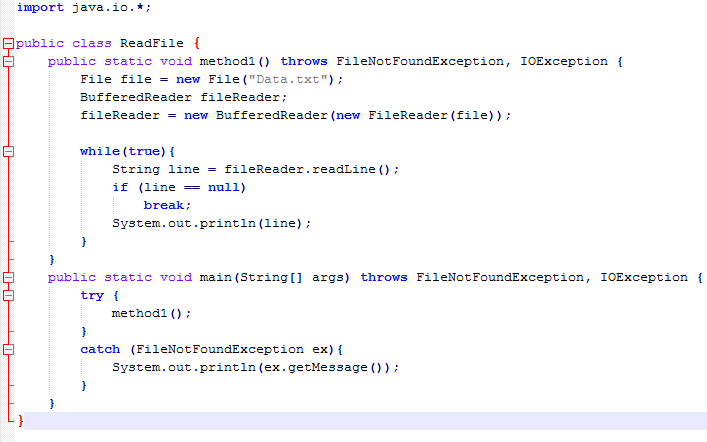


Keyword *throw* bisa digunakan untuk checked dan unchecked exception, namun untuk contoh di atas *throw* yang digunakan adalah untuk checked exception karena exceptionnya FileNotFoundException yang bukan merupakan subclass dari java.lang. RuntimeException.

Tugas:

1. Terdapat dua cara untuk menangani Exception yaitu dengan menangkap Exception dan melempar Exception. Lakukan penanganan exception dengan menangkap Exception menggunakan blok try-catch. Berilah penjelasan (apakah program termasuk unchecked exceptions atau checked exceptions)!





Keyword *try-catch* bisa digunakan untuk checked dan unchecked exception, namun untuk contoh di atas *try-catch* yang digunakan adalah untuk checked exception karena exceptionnya masih bisa dihandle.

1. Buatlah sebuah class **Stack, FullStackException dan EmptyStackException**. Class Stack ini menggambar Stack yang menerapkan konsep LIFO (Last In First Out). Konsep LIFO ini, data yang terakhir masuk akan keluar pertama kali.

