CyberSecurity dan Forensik Digital e-ISSN: 2615-8442
Vol. 4, No. 1, Mei 2021, hlm. 13-17

SECURE CODE DESIGN PADA PEMROGRAMAN BERORIENTASI OBJEK DENGAN PENANGANAN PENGECUALIAN

Rahmawati Nafi'ah

Magister Informatika, Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
rahmawati.nafiah@gmail.com

(Naskah masuk: 1 Februari 2021, diterima untuk diterbitkan: 31 Mei 2021)

Abstrak

Proses pengembangan perangkat lunak harus mengikuti tahapan tertentu yang disebut dengan *Software Development Life Cycle* atau (SDLC). Pada pengembangan perangkat lunak, yang belum nampak secara eksplisit pada conventional SDLC adalah aspek keamanan. Keamanan seharusnya hadir pada setiap tahapan SDLC. Keamanan perangkat lunak bisa dimulai dari *security requirement, secure design, secure coding*, hingga pengujian. Tahapan *coding* merupakakan implementasi dari desain dalam bentuk kode. Programmer harus berhati-hati agar tidak ada lubang keamanan pada saat perangkat lunak dikembangkan. Membuat perangkat lunak yang aman dengan desain memerlukan pertimbangan pada bagiamana cara menangani kesalahan, terutama pada tahapan *coding*. Bahasa pemrograman Java yang memiliki sifat mengurangi kemungkinan terjadinya kesalahan tipe data. Bahasa termasuk ke dalam pemrograman berorientasi objek. Pemrograman berorientasi objek merupakan teknik membuat suatu program berdasarkan objek dan hal yang bisa dilakukan oleh objek tersebut. Bahasa Java menyediakan fitur penanganan pengecualian, seperti pernyataan *throw* dan blok *try-catch-finally*. Pada bahasa ini terdapat *exception handling* yaitu mekanisme penangan error yang mungkin terjadi dalam suatu program.

Kata kunci: *secure coding, pemrograman berorientasi objek, exception handling*

SECURE CODE DESIGN ON OBJECT ORIENTED PROGRAMMING WITH EXCEPTION HANDLING

Abstract

The software development process must follow certain stages called the Software Development Life Cycle (SDLC). In software development, what has not been explicitly seen in conventional SDLC is the security aspect. Security should be present at every stage of SDLC. Software security can be started from security requirements, secure design, secure coding, to testing. The coding stage is the implementation of the design in code form. Programmers must be careful that there are no security holes when software is being developed. Creating safe software by design requires consideration of how to handle errors, especially at the coding stage. The Java programming language has the property of reducing the possibility of data type errors. This language belongs to object-oriented programming. Object-oriented programming is a technique of creating a program based on objects and what these objects can do. The Java language provides exception handling features, such as throw statements and try-catch-finally blocks. In this language, there is exception handling, which is an error handling mechanism that may occur in a program..

Keywords: *secure coding, object oriented programing, exception handling*

1. PENDAHULUAN

Proses pengembangan perangkat lunak harus mengikuti tahapan tertentu yang disebut dengan *Software Development Life Cycle* atau (SDLC). SDLC merupakan proses mengembangkan atau merubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan best practice atau cara-cara yang sudah teruji dengan baik(S. & Shalahuddin, 2016).

Secara umum tahapannya meliputi requirement, desain, implementasi (coding), pengujian hingga peluncuran (deployment). Dikutip dari (Rahardjo, 2014) hal yang belum nampak secara eksplisit pada conventional SDLC adalah aspek keamanan. Keamanan seharusnya hadir pada setiap tahapan SDLC.

Masalah keamanan pada perangkat lunak yang sudah dikembangkan biasanya dilakukan pada tahapan pengujuan. Tahapan ini biasanya dilakukan setelah perangkat lunak dikembangkan. Namun pengujian pada tahapan ini bisa dikatakan sudah terlambat karena membutuhkan biaya yang besar pada proses perbaikannya jika dibandingkan dengan memperhatikan aspek keamanan pada setip tahapan pengembangnnya.

Keamanan perangkat lunak bisa dimulai dari *security requirement*, *secure design*, *secure coding*, hingga pengujian. Semua tahapan dibuat mengacu pada aspek keamanan yaitu CIA (*Confidentiality, Integrity, Availability*). Pada jurnal ini membahas mengenai desain dari tahapan *secure coding*. Tahapan ini merupakakan implementasi dari desain dalam bentuk kode. Programmer harus berhati-hati agar tidak ada celah/ lubang keamanan pada saat perangkat lunak dikembangkan. Membuat perangkat lunak yang aman dengan desain memerlukan pertimbangan pada bagiamana cara menangani kesalahan, terutama pada tahapan *coding*.

Ada bahasa pemrograman yang membatasi gerak pemrogram sehingga dapat mengurangi kesalahan yang mungkin terjadi. Sebagai contoh, bahasa pemrograman yang memiliki sifat *strongly typed* (seperti bahasa Java) mengurangi kemungkinan terjadinya kesalahan tipe data. Namun, kesalahan masih dapat terjadi dari sisi lainnya (Rahardjo, 2014).

Bahasa java merupakan salah satu bahasa yang mendukung pemrograman berorientasi objek. Pemrograman berorientasi objek adalah teknik membuat suatu program berdasarkan objek dan hal yang bisa dilakukan oleh objek tersebut. Menurut (Purohit & Tokekar, 2018), bahasa untuk pengembangan perangkat lunak harus menyediakan bagi para pengembang fungsi dan fitur yang membuatnya nyaman untuk menghadapi keadaan yang tidak normal dan tidak menyenangkan, dan juga untuk memulihkan kesalahan, untuk membuat produk perangkat lunak akhir menjadi kuat.

Bahasa pemrograman modern, seperti Java dan C #, biasanya menyediakan fitur penanganan pengecualian, seperti pernyataan throw dan blok try-catch-finally. Fitur-fitur ini memisahkan kode penanganan kesalahan dari kode sumber biasa dan dalam praktiknya dimanfaatkan secara luas untuk mendukung pemahaman dan pemeliharaan perangkat lunak (Melliar-Smith & Randell, 1985)(Chen et al., 2009) (de Pádua & Shang, 2017). Pustaka penelitian ini berdasarkan studi literatur naskah-naskah penelitian terkait *exception handling* pada pemrograman berorientasi objek yaitu bahasa java.

2. DASAR TEORI

2.1. Kesalahan Pada Program

Pada proses pengembangan perangkat lunak, terkadang terjadi kesalahan atau error yang disebabkan oleh banyak hal. Mernurut (Slamet & Suhartanto, 1992) dalam (Fauziah et al., 2019), penanganan kesalahan (error handling) dilakukan bila terjadi kesalahan dalam penulisan program sumber, baik kesalahan penulisan besaran leksikal, kesalahan sintaksis, maupun kesalahan semantik.

Pada pemrograman java terdapat tiga kategori *error* yaitu *syntax error, logical error* dan *runtime error*. Hal itu sesuai dengan model Miles & Huberman pada jurnal Analisis Kesalahan Coding Bahasa Pemrograman Java Pada Matakuliah Algoritma Pemrograman Mahasiswa Tadris Matematika IAIN Kediri oleh (Syamsudin, 2020). Menurut penelitian (Hasanah & Untari, 2018) jika dilihat dari urutan taraf kesukaran pada saat mahasiswa menganalisis error code program yang terjadi *syntax error* ini tergolong yang paling mudah dideteksi, selanjutnya *Run-time error* dan yang paling susah adalah ketika menjumpai kesalahan jenis *logical error*. Berikut ini pengetian *syntax error, logical error* dan *runtime error* dalam buku Pemrograman Berorientasi Objek compiler (Lita Likmalatri, 2016).

2.1.1 *Syntax error*

Syntax error adalah sebuah *error* yang terjadi karena ada aturan bahasa pemrograman yang tidak ditaati, error ini akan dideteksi oleh.

2.1.2 *Logical error*

Logical error adalah sebuah error yang terjadi ketika program bekerja tidak sesuai dengan tujuan dari pembuatan program itu. Error jenis ini ditangani dengan *debugging* .

2.1.3 *Runtime error*

Runtime error merupakan gangguan atau masalah yang terjadi ketika sebuah program computer dijalankan. Hal ini bisa diatasi dengan restart computer dari awal lagi. Namun, ketika error terus menerus muncul, hal yang harus dilakukan dengan mengecek pada system operasi.

2.2. *Exception*

Menurut (Lita Likmalatri, 2016) *exception* adalah kondisi yang akan muncul, jika suatu program tidak sukses dijalankan, atau dengan kata lain , user tidak mengisi input sesuai dengan syarat yang berlaku. Pada java terdapat dua kategori besar untuk *exception* yaitu *checked exception* dan *unchecked exception*.

2.3. Exception Handling

Exception Handling adalah sebuah konteks eksekusi dari kejadian yang sebenarnya untuk menangani kesalahan dengan pengecualian yang ditetapkan dan dijelaskan waktu mulai, waktu akhir dan status yang di dapat (Brambilla et al., 2006) (Fahrudin & Majapahit, 2018). Menurut (Ilham & Naziro, 2020), *exception handling* adalah event yang terjadi ketika program menemui kesalahan pada saat instruksi program dijalankan.

Untuk mengimplementasikan exception handling digunakan lima keyword yaitu *try*, *catch*, *finaly*, *throw* dan *throws*. Pada buku Pemrograman Berorientasi Objek compiler (Lita Likmalatri, 2016) diejlaskan pengertian dari kelima *keyword* tersebut.

2.3.1 *Try Catch*

Blok statement *try* membingkai kode yang kemungkinan akan memunculkan exception dan mendefinisikan exception handler yang dapat menanganinya. Blok *catch* digunakan untuk menempatkan kode-kode program java yang digunakn untuk menangani *exception* tertentu.

2.3.2 Blok *Finaly*

Keyword *finnaly* merupakan *keyword* yang menunjukkan bahwa blok program tersebut akan selalu dieksekusi meskipun ada kesalahan yang muncul ataupun tidak ada.

2.3.3 *Throw*

Keyword *throw* digunakan untuk melemparkan suatu bug yang dibuat secara manual.

2.3.4 *Throws*

Jika method tidak dapat menangani exception, maka method tersebut harus mendeklarasikan jenis *exception* yang mungkin dilemparkan dengan kata *throws*. *Keyword throws* akan diikuti daftar tipe exception yang mungkin dilempar oleh sebuah method.

3. PEMBAHASAN

Pada bagian ini akan dibahas mengenai beberapa kode program dalam kategori uncheked exception dan penanganannya. Kode program ini ditulis menggunakan bahasa java menggunakan software NetBeans IDE. Semua program ditulis menggunakan package exception. Package merupakan pengelompokan kelas-kelas berdasarkan kesamaan atau kemiripan fungsi pada pemrograman java.

Kata public memnadakan bahwa *acces modifier* nya bersifat publik. Semua method dan variabelnya bisa diwariskan oleh semua subclassnya dan dapat diakses dimanapun. Kata static membuat program dapat mengakses atruibut/method dari kelas secara langsung tanpa harus membuat objek. Main menandakan bahwa method tersebut adalah method utama. System.out.println merupakan perintah untuk menampilkan pesan pada java dilayar.

3.1. *Null Pointer Exception*

NullException119.java

```
1  package exception;
2  public class NullException119 {
3  public static void main (String []args){
4       String s = null;
5      try {
6          s.length();
7      }
8      catch (NullPointerException e){
9          System.out.println("ada error yang terjadi");
10     }
11 }
12 }
```

Gambar 1. Null Pointer Exception

Null Pointer Exception merupakan penggunaan NULL yang tidak sah. Pada program nama kelas yang digunakan adalah NullException sesuai dengan exception yang dibahas. Pada program dibuat variabel baru bernama s yang tidak memiliki nilai. Jika exception dengan blok try yang bersisikan s.length maka akan otomatis dilempar pada blok catch.

Pada blok try, method length() digunakan untuk mengetahui panjang objek. Karena variabel s tidak memiliki nilai otomatis nialinya adalah NULL. Karena kosong maka program akan menjalankan blok catch sehingga didapatkan pesan kesalahan yaitu "ada error yang terjadi".

```
run:
ada error yang terjadi
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 2. Hasil program Null Pointer Exception

3.2. *Aritmethic Exception*

ArithmeticException119.java

```
1  package exception;
2  public class ArithmeticException119 {
3  public static void main (String [] args){
4      int a = 3;
5      int b = 0;
6       try {
7          System.out.println(a/b);
8      }
9      catch (ArithmeticException e){
10         System.out.println("ada error yang terjadi");
11     }
12 }
13 }
14
```

Gambar 3. *Aritmethic Exception*

Aritmethic Exception merupakan kesalahan aritmetik seperti pembagiaan dengan nol. Pada kode program dideklarasikan variabel a dan b dengan tipe data integer. Variabel a diberikan nilai 3 dan variabel b diberikan nilai 0.

Pada blok Try dilakukan operasi pembagian yaitu dengan membagi variabel a dengan b. Kerena nilai pembagian dengan 0 termasuk aritmethic exception maka program akan dilemparkan ke blok catch. Maka program yang dihasilkan akan menampilkan pesan kesalahan.

3.3. *Array Index Out Of Bounds Exception*

ArrayIndexOutOfBoundsException119.java

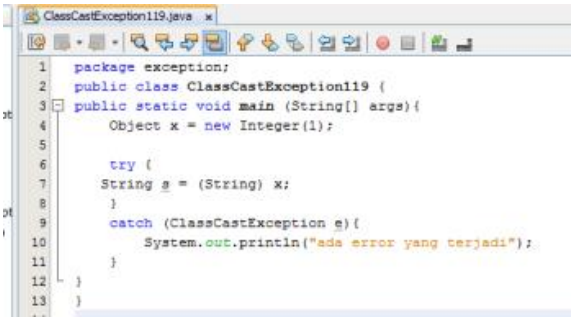
```
1  package exception;
2  public class ArrayIndexOutOfBoundsException119 {
3  public static void main (String [] args){
4  int[] a= new int[10];
5
6       try {
7        a[10]=11;
8      }
9      catch (ArrayIndexOutOfBoundsException e){
10         System.out.println("ada error yang terjadi");
11     }
12 }
13 }
```

Gambar 4. *Array Index Out Of Bounds Exception*

Exception ini menyatakan bahwa indeks array ada diluar batas. Exception ini digunakan pada tipe data array. Pada kode program dibuat data bari dengan tipe data integer dengan jumlah 10. Index pada tipe data array dimulai dari), Karena jumlah datanya adalah 10, maka index terakhir adalah 9.

Pada blok try , program mencoba memberikan nilai pada index 10 yaitu dengan nilai 11. Saat program dijalankan, program akan mengecek jumlah index. Karena jumlah index yang ditunjuk ternyata melebihi yang index yang tersedia maka program akan melemparkan ke blok catch untuk memdapatkan pesan error.

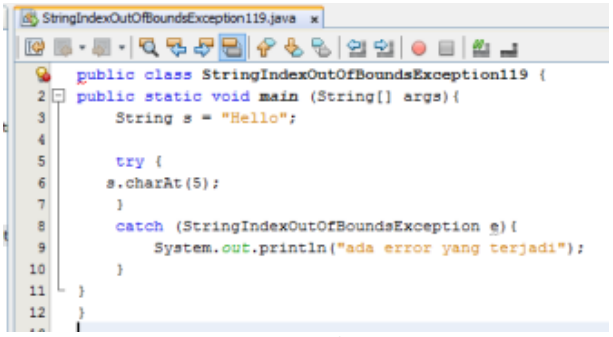
3.4. *Class Cast Exception*



Gambar 5. *Class Cast Exception*

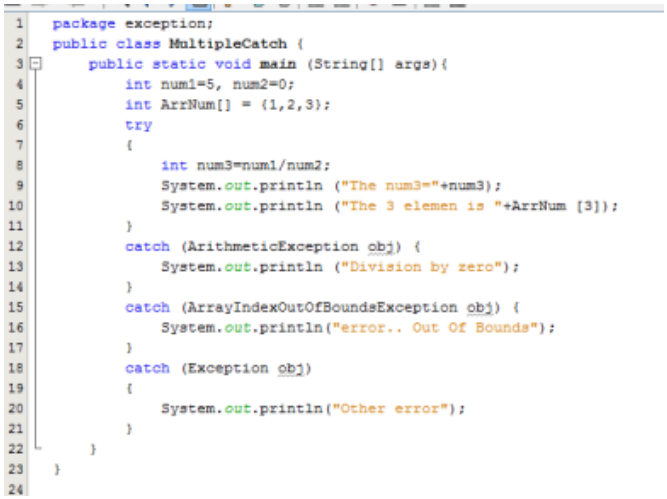
Class Cast merupakan cast yang tidak sah dalam program. Pada program diatas dideklarasikan objek dengan nama x dengan tipe data integer. Namun pada blok try, program ingin mengakses objek dengan memberikan tipe data string. Karena kedua tipe data berbeda maka otomatis program akan melempar ke blok catch dan akan memberikan pesan kesalahan yang ada.

3.5. String Index Out Of Bounds Exception



Gambar 6. *String Index Out Of Bounds Exception*

String Index Out Of Bounds merupakan exception karenaindekx berada diluar batas string. Pada program dibuat variabel baru dengan tipe data string dengan nama s. Variabel s kemudian diberikan data berupa kata "Hello". Pada blok try variabel s diberikan method charAt untuk mengetahui posisi karaketer pada indek tertentu. Indek yang dimaksud adalah indek 5. Pada program java index dimulai dari 0, dan variebel s memiliki 5 karakter yang berarti indek terkahir adalah 4. Ketika blok try menciba mengakses indek 5, yang tidak ada pada varibel maka otomatis akan dilempar ke blok catch untuk menampilkan pesan kesalahan yang ada.



Gambar 7. *Multiple catch Exception*

Pada gambar 7 diberikan contoh exception penggabungan *aritmethic exception* dengan *array index out of bounds.* Pada blok try diberikan 3 perintah. Pada setiap baris akan dieksekusi oleh program dan akan dicek hasilnya. Pada baris ke 8, sudah ada kesalahan pembagian dengan 0 maka program akan berhenti dan menampilkan kesalahan aritmethicException yaitu *Division by zero*.

4. PENUTUP

Pada tahapan pembuatan program pasti ada bagian program yang tidak sempurna dan tidak ada pengguna program yang sempurna. Diperlukan mekanisme untuk mengani error atau kesalahan yang terjadi. Bahasa java menyediakan mekanisme penangan tersebut dengan exception. Exception menandakan bahwa ada event yang tidak biasa yang mengganggu alur instruksi. Sehinga bisa dikatakan bahwa exception termasuk dalam runtime error. Oleh karena itu diperlukan penanganan kesalahan.

DAFTAR PUSTAKA

BRAMBILLA, M., COMAI, S., & TZIVISKOU, C. (2006). Exception Management Within Web Applications Implementing Business Processes. *Springer, Berlin, Heidelberg*, *4119*. https://doi.org/https://doi.org/10.1007/11818502_6

CHEN, C.-T., CHENG, Y. C., HSIEH, C.-Y., & WU, I.-L. (2009). Exception handling refactorings: Directed by goals and driven by bug fixing. *Journal of Systems and Software*, *82*(2), 333–345.

DE PÁDUA, G. B., & SHANG, W. (2017). Revisiting exception handling practices with exception flow analysis. *ArXiv*, *August*.

FAHRUDIN, & MAJAPAHIT, S. A. (2018).

Perancangan Exception Handling Pada Sistem Layanan Sidang Tugas Akhir. *Knsi 2018*, 8–9.

FAUZIAH, F., APRIANSYAH, A., SAPUTRA, T. I., & WIJAYA, Y. F. (2019). Desain Mesin Compiler untuk Penganalisa Leksikal, Sintaksis, Semantik, Kode Antara dan Error Handling Pada Bahasa Pemrograman Sederhana. *Journal of Applied Informatics and Computing*, *3*(1), 01–07. https://doi.org/10.30871/jaic.v3i1.1153

HASANAH, F. N., & UNTARI, R. S. (2018). Analisis Kemampuan Mendeteksi Eror Kode Program Mata Kuliah Pemrograman Berorientasi Objek Pada Program Studi Pendidikan Teknologi Informasi Universitas Muhammadiyah Sidoarjo. *Teknologi Dan Kejuruan: Jurnal Teknologi, Kejuruan, Dan Pengajarannya*, *41*(2), 139–146. https://doi.org/10.17977/um031v41i22018p139

ILHAM, N. A., & NAZIRO. (2020). Implementasi Konsep Pemrograman Berorientasi Objek Pada Aplikasi Sistem Parkir Menggunakan Bahasa Pemrograman Java. *Jurnal Edukasi Elektro*, *3*(2), 63–69. https://doi.org/10.21831/jee.v3i2.28293

LITA LIKMALATRI. (2016). *Pemrograman Berorientasi Objek* (C. C. Dewi (ed.)). Putra Nugraha.

MELLIAR-SMITH, P. M., & RANDELL, B. (1985). Software Reliability: The Role of Programmed Exception Handling. *Reliable Computer Systems*, 143–153.

PUROHIT, P., & TOKEKAR, V. (2018). *An Investigation of Exception Handling Practices in . Net and Java Environments*. *13*(5), 2130–2140.

RAHARDJO, B. (2014). *Keamanan Perangkat Lunak*. 29. http://budi.rahardjo.id/files/software-security.pdf

S., R. A., & SHALAHUDDIN, M. (2016). *Rekayasa Perangkat Lunak*. Informatika.

SLAMET, S., & SUHARTANTO, H. (1992). *Teknik Kompilasi*. Universitas Indonesia.

SYAMSUDIN, A. . (2020). Analisis Kesalahan Coding Pemrograman Java Pada Matakuliah Algoritma Pemrograman Mahasiswa Tadris Matematika Iain Kediri. *Factor M*, *2*(2), 102–114. https://doi.org/10.30762/f_m.v2i2.1711