# Sistem Penilaian Otomatis untuk Kode Program Algoritma dan Struktur Data Menggunakan *Process Builder*

**Sugiarto Cokrowibowo*[1], Nuralamsah Zulkarnaim[2], Muh. Fahmi Rustan[3], Dita Yustianisa[4]**
[1,2,3,4] Program Studi Informatika, Universitas Sulawesi Barat
E-mail: *[1]sugiarto.cokrowibowo@unsulbar.ac.id, [2]nuralamsah@unsulbar.ac.id,
[3]muhfahmi@unsulbar.ac.id, [4]yustianisadita@gmail.com

***Abstract***

*As the number of students in the computer science or informatics program increases, there is a pressing need for innovative teaching strategies that focus on preparing graduates for careers in the field of computer science, where one of the core competencies is the ability to write code effectively. To enhance students' programming competencies, it is essential to provide them with ample practice in writing program code so that they can gain sufficient experience to solve computational problems. Given the high number of programming exercises assigned by instructors and the disproportionate ratio of students to teachers, the implementation of an automated program evaluation system is highly necessary, particularly for courses in algorithms and data structures. This automated assessment system should be objective, fair, and capable of providing feedback to students. In this paper, we propose an automated assessment system using Java ProcessBuilder. This automated assessment system operates based on the black-box testing principle. Java ProcessBuilder is capable of building an automated testing system to assess program code and provide feedback, including correct output and identifying program errors that students can rectify.*

***Keywords****—automated assessment, process builder, black box testing*

***Abstrak***

*Seiring peningkatan jumlah mahasiswa peserta didik pada program studi ilmu komputer atau informatika maka sangat diperlukan inovasi dan strategi pembelajaran yang lebih focus pada mempersiapkan lulusan untuk berkarir di bidang ilmu komputer dimana salahsatu kompetensi utamanya adalah kemampuan menulis kode program secara efektif. Untuk meningkatkan kompetensi pemrograman mahasiswa sangat dibutuhkan banyak latihan menulis kode program sehingga mahasiswa dapat memiliki pengalaman yang cukup untuk menyelesaikan permasalahan komputasi. Banyaknya jumlah latihan menulis program yang diberikan oleh pengajar disamping pengikatan jumlah mahasiswa yang tidak berbanding lurus dengan peningkatan jumlah pengajar maka sangat dibutuhkan adanya sistem penilaian otomatis untuk kode program khususnya pada mata kuliah algoritma dan struktur data. Sistem penilaian otomatis ini harus bersifat objektif, adil, serta mampu memberikan umpan balik kepada peserta didik. Pada tulisan ini diajukan sebuah sistem penilaian otomatis menggunakan Java ProcessBuilder. Sistem penilaian otomatis ini bekerja berdasarkan prinsip pengujian black box. Java ProcessBuilder mampu untuk digunakan dalam membangun sistem pengujian otomatis untuk menguji kode program dan memberikan umpan balik hasil baik berupa output yang benar maupun kesalahan program yang dapat kembali diperbaiki oleh peserta didik.*

***Kata Kunci****—sistem penilaian otomatis, process builder, pengujian black box*

## 1. PENDAHULUAN

Seiring dengan kemajuan disiplin ilmu komputer, para pendidik juga menerapkan inovasi metode dan strategi pembelajaran yang memiliki focus lebih besar berdasarkan pengalaman untuk mempersiapkan lulusan dalam menghadapi karir komputasi. Prinsip utama yang mendasari dan menjadi kemampuan inti dalam disiplin ilmu komputer adalah penulisan kode pemrograman yang efektif, karena hal ini akan membantu menunjukkan bakat berfikir komputasional tingkat tinggi, logika, penalaran algoritmik, dekomposisi masalah, iterasi, rekursi dansecara keseluruhan berkaitan dengan penanganan kompleksitas kode program[1]. Algoritma dan struktur data merupakan dasar dari ilmu komputer dan rekayasa perangkat lunak hal ini disebabkan karena setiap teori komputasional dan program dunia nyata terdiri dari algoritma yang dioperasikan pada elemen-elemen data yang memiliki struktur dasar. Kemampuan dalam memilih solusi komputasi yang tepat untuk permasalahan dunia nyata memerlukan pemahaman kemampuan teoritis dan praktis tentang algoritma dan struktur data[2]. Secara umum pengajaran pemrograman sangatlah menantang [3], dan hanya sedikit informasi tentang bagaimana cara untuk meningkatkan kemahiran programmer pemula [4]. Menulis kode program merupakan keahlian yang terus berkembang seiring dengan frekuensi latihan yang dikerjakan oleh programmer. Programmer harus memiliki pemahaman yang baik tentang sintaksis kode, struktur program, ekspresi logika, struktur data, dan algoritma yang kesemuanya itu memberikan kontribusi pada peningkatan kemampuan penulisan kode program yang efektif. Oleh karena itu pengajar berupaya meningkatkan pembelajaran pemrograman kepada mahasiswa ilmu komputer dengan mengintegrasikan berbagai variasi skenario pemrograman yang diwujudkan dalam beberapa konsep pemrograman dalam disiplin ilmu komputer. Latihan pemrograman dengan berbagai tingkat kesulitan serta frekuensi pengulangan latihan akan meningkatkan kemampuan pemrograman mahasiswa. Namun, pengajar seringkali menemui kesulitan dalam memberikan umpan balik individual, yang mungkin disebabkan oleh kurangnya waktu maupun jumlah siswa yang besar[5]. Selain itu evaluasi secara manual terhadap tugas kode program yang diserahkan oleh mahasiswa membutuhkan pekerjaan berulang dan memakan waktu sehingga memberikan beban tambahan kepada tenaga pengajar. Penilaian oleh tenaga pengajar berbeda mungkin juga menerapkan kriteria penilaian yang berbeda. Oleh karena itu penilaian secara manual oleh pengajar berbeda bersifat subjektif dan memiliki kemungkinan inkonsistensi yang menjadi ketidak adilan bagi mahasiswa [6]. Dengan alasan ini untuk meningkatkan objektivitas dan konsistensi maka diperlukan adanya suatu sistem penilaian otomatis untuk kode pemrograman.

Perangkat penilaian otomatis dikembangkan untuk membantu pengajar dalam memberikan penilaian atas pekerjaan tugas kuliah dari mahasiswa serta memberikan umpan balik kepada mahasiswa[7]. Studi ini memperluas pengembangan sistem penilaian otomatis khusus untuk mata kuliah pemrograman yang mengimplementasikan algoritma dan struktur data.

## 2. METODE

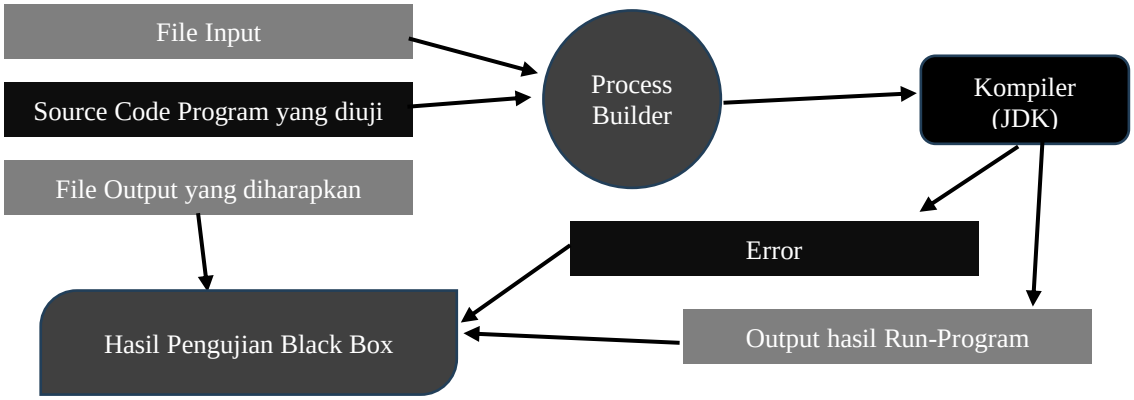
Untuk mengevaluasi dan memberikan umpan balik tugas pemrograman yang diberikan kepada mahasiswa maka perlu dirancang sebuah system pengujian otomatis terhadap kode program tersebut. Sistem ini harus dikembangkan dengan mengedepankan otomatisasi dalam penilaian tugas, pemberian umpan balik, objektivitas dan keadilan. Sitem penilaian otomatis ini dikembangkan menggunakan pustaka *ProcessBuilder* di Java. *ProcessBuilder* merupakan pustaka (*class*) yang tersedia di *Java Development Kit* (JDK) yang digunakan untuk membuat proses pada system operasi. Pada pengembangan sistem penilaian otomatis untuk kode program algoritma dan struktur data pustaka *ProcessBuilder* akan bertugas untuk menjalankan terminal system operasi yang kemudian akan menjalankan compiler, membaca input dan menuliskan output program. Selanjutnya output program akan diperiksa kesesuaiannya dengan output

program yang seharusnya menggunakan mekanisme pengujian *black box*. Pengujian *black box* ini telah digunakan di berbagai software pengujian yang tujuan utamanya adalah untuk menguji fungsionalitas program serta perilakunya terhadap input dan output[8]. Mekanisme pengujian *black box* ini dipilih dan sangat sesuai untuk digunakan dalam mengembangkan sistem penilaian otomatis pada tugas pemrograman dikarenakan pada dasarnya tugas pemrograman sifatnya akan sangat terbuka, sebuah masalah pemrograman akan memiliki banyak varian solusi yang akan tidak sesuai jika dilakukan pengujian menggunakan *string matching* sederhana.

Metode pengujian *black box* yang digunakan pada sistem penilaian otomatis ini akan memeriksa kesesuaian antara output program yang dihasilkan dengan output program yang diharapkan Ketika diberikan input. Metode pengujian *black box* akan dijalankan oleh *ProcessBuilder*. *ProcessBuilder* pertama akan membaca input dan output yang diharapkan, kemudian *ProcessBuilder* akan menjalankan program yang diuji. Program yang diuji akan menghasilkan output. Output yang dihasilkan tersebut akan dipersiksa kesesuaiannya dengan output yang diharapkan menggunakan operasi *string matching*. Jika output yang dihasilkan sama dengan output yang diharapkan maka program yang diuji akan dinyatakan benar (*correct*). Sebaliknya, Jika output yang dihasilkan tidak sama dengan output yang diharapkan maka program yang diuji akan dinyatakan salah (*wrong answer*). Jika terjadi error atau kesalahan program makan akan diberikan jenis kesalahan program yang terjadi.

### *2.1 Pengembangan Sistem Pemeriksaan Otomatis*

Sistem ini dikembangkan menggunakan bahasa pemrograman java yang didukung oleh pustaka *ProcessBuilder*. *ProcessBuilder* akan menjalankan *process* pada system operasi yang akan memanggil kompiler.




Gambar 1 Tahapan penilaian otomatis

*Pseudocode* untuk sistem penilaian otomatis menggunakan *ProcessBuilder* diperlihatkan sebagai berikut:

```
//-------------------------------------------------------------------------------------------------------------
BEGIN
   sDir ← "direktori sumber daya pengujian (resource)"
   sSourceCode ← "Matriks.java"
   sInput ← "input.txt"
   sOutput ← "output.txt" // output yang seharusnya

   TRY
      input ← Baca isi file di lokasi sDir + sInput
      output ← Baca isi file di lokasi sDir + sOutput
```

```
    processBuilder ← Buat objek ProcessBuilder
    processBuilder.directory ← Set lokasi direktori kerja ke File(sDir)
    processBuilder.command ←Set perintah eksekusi ke "cmd.exe", "/c", "java " + sSourceCode
    processBuilder.redirectInput ← Set input dari file di sDir + sInput

    process ← Mulai proses sesuai processBuilder
    process.TungguSelesai()

    // Hasil Pengujian
    outputTesting ← Ekstrak output dari proses
    errorTesting ← Ekstrak pesan kesalahan dari proses

    IF errorTesting.length() > 0 THEN
      Cetak "RESULT:"
      Cetak "-----------------------------------------"
      Cetak "Ditemukan Error Program"
      Cetak errorTesting
      Cetak "-----------------------------------------"
    ELSE
      Cetak "RESULT:"
      Cetak "-----------------------------------------"
      Cetak "Program Berhasil"
      Cetak outputTesting
      Cetak "-----------------------------------------"
      IF output.trim().equals(outputTesting.trim()) THEN
        Cetak "CORRECT"
      ELSE
        Cetak "WRONG ANSWER"
      END IF
    END IF
  CATCH Exception AS ex
    Cetak ex.stackTrace()
  END TRY
END
//-------------------------------------------------------------------------------------------------------------
```

## 3. HASIL DAN PEMBAHASAN

Mekanisme pengujian program menggunakan *ProcessBuilder* dimulai dari pembacaan file-file sumberdaya berikut:

1. File kode sumber yang akan diuji (*source code* berkestensi .java)
2. File *text* berisi input yang dibutuhkan oleh *source code* saat dijalankan
3. File *text* berisi output yang seharusnya jika *source code* berhasil ditulis dengan benar.

Langkah selanjutnya ialah membuat instan dari *class ProcessBuilder* (*ProcessBuilder processBuilder = new ProcessBuilder( );* ). *Object* dari *class ProcessBuilder* ini selanjutnya yang akan digunakan untuk menjalankan *process* pada sistem operasi.

```
//-------------------------------------------------------------------------------------------------------------
package testing;

import java.io.BufferedReader;
```

```
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Path;

public class Tester {

   public static void main(String[] args) {
      String sDir =
"C:/Users/ASUS/OneDrive/Documents/NetBeansProjects/PengujianSourceCode/src/testing/";
      String sSourceCode = "Matriks.java";
      String sInput = "input.txt";
      String sOutput = "output.txt";//output yang seharusnya

      try {
         String input = Files.readString(Path.of(sDir + sInput));
         String output = Files.readString(Path.of(sDir + sOutput));

         ProcessBuilder processBuilder = new ProcessBuilder();
         processBuilder.directory(new File(sDir));
         processBuilder.command("cmd.exe", "/c", "java " + sSourceCode);
         processBuilder.redirectInput(new File(sDir + sInput));

         Process process = processBuilder.start();
         process.waitFor();

         //RESULT
         String outputTesting = extractOutput(process);
         String errorTesting = extractError(process);

         if (errorTesting.length() > 0) {
            System.out.println("RESULT:");
            System.out.println("-----------------------------------------");
            System.err.println("Ditemukan Error Program");
            System.out.println(errorTesting);
            System.out.println("-----------------------------------------");
         } else {
            System.out.println("RESULT:");
            System.out.println("-----------------------------------------");
            System.out.println("Program Berhasil");
            System.out.println(outputTesting);
            System.out.println("-----------------------------------------");
            if (output.trim().equals(outputTesting.trim())) {
               System.out.println("CORRECT");
            }else{
               System.err.println("WRONG ANSWER");
            }
         }
      } catch (Exception ex) {
         ex.printStackTrace();
      }
```

```
    }

    public static String extractOutput(Process process) throws IOException {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(process.getInputStream()));
        StringBuffer sb = new StringBuffer();
        String line;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        return sb.toString();
    }

    public static String extractError(Process process) throws IOException {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(process.getErrorStream()));
        StringBuffer sb = new StringBuffer();
        String line;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        return sb.toString();
    }

}
//-------------------------------------------------------------------------------------------------------------
```

Pengujian untuk *source code* **Matriks.java** dengan input output berikut:

Input

```
2 4
```

Output

```
1       2       3       4
5       6       7       8
```

Menghasilkan output pengujian sebagai berikut:

```
RESULT:
-----------------------------------------
Program Berhasil
1       2       3       4
5       6       7       8


-----------------------------------------
CORRECT
```

## 4. KESIMPULAN

Sistem penilaian tugas pemrograman otomatis ini sangat membantu pengajar dalam melakukan penilaian tugas pemrograman untuk mata kuliah algoritma dan struktur data. Sistem ini memiliki kemampuan untuk menilai tugas mahasiswa dalam jumlah besar secara konsisten

dan memberikan umpan balik kepada mahasiswa. Untuk mengembangan lebih lanjut pada studi berikutnya akan dikembangkan sistem penilaian otomatis yang dapat mengakomodir berbagai baha pemrograman karena dalam konsep algoritma dan struktur data sebenarnya yang dikembangkan adalah sistematika berfikir logisnya yang seharusnya dapat diterapkan bukan hanya pada satu bahasa pemrograman sehingga sistem pengujian yang dibangun juga harus dapat mengakomodir jika mahasiswa mengerjakan tugas algoritma dan struktur data menggunakan berbagai bahasa pemrograman yang berbeda-beda.

## REFERENSI

[1] D. John Lemay, R. B. Basnet, T. Doleck, P. Bazelais, and A. Saxena, "Instructional interventions for computational thinking: Examining the link between computational thinking and academic performance," *Comput. Educ. Open*, vol. 2, p. 100056, 2021, doi: 10.1016/j.caeo.2021.100056.

[2] The Joint Task Force on Computing Curricula, "Computer Science Curricula 2023," no. March, pp. 17–35, 2023.

[3] T. Daradoumis, J. M. Marquès Puig, M. Arguedas, and L. Calvet Liñan, "Analyzing students' perceptions to improve the design of an automated assessment tool in online distributed programming," *Comput. Educ.*, vol. 128, pp. 159–170, 2019, doi: 10.1016/j.compedu.2018.09.021.

[4] A. Luxton-Reilly *et al.*, *Introductory programming: A systematic literature review*. 2018. doi: 10.1145/3293881.3295779.

[5] R. P. Medeiros, G. L. Ramalho, and T. P. Falcao, "A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education," *IEEE Trans. Educ.*, vol. 62, no. 2, pp. 77–90, 2019, doi: 10.1109/TE.2018.2864133.

[6] D. Insa and J. Silva, "Automatic assessment of Java code," *Comput. Lang. Syst. Struct.*, vol. 53, pp. 59–72, 2018, doi: 10.1016/j.cl.2018.01.004.

[7] J. Skalka and M. Drlik, "Automated assessment and microlearning units as predictors of at-risk students and students' outcomes in the introductory programming courses," *Appl. Sci.*, vol. 10, no. 13, 2020, doi: 10.3390/app10134566.

[8] L. M. Castro and M. A. Francisco, "A language-independent approach to black-box testing using Erlang as test specification language," *J. Syst. Softw.*, vol. 86, no. 12, pp. 3109–3122, 2013, doi: 10.1016/j.jss.2013.07.021.