PBO - Modul Praktik 5

Pada modul ini kita akan menggunakan library Apache Common CSV untuk penyimpanan data password pada file CSV serta load data Password dari CSV.

Penambahan Dependency

Penambahan dependency pada Gradle cukup dengan menambahkan koding berikut pada file `build.gradle` di bagian `dependencies`:

```
implementation 'org.apache.commons:commons-csv:1.10.0'
```

Sedangkan penambahan dependency pada Maven dilakukan dengan menambahkan block XML berikut pada file `pom.xml` di dalam block `<dependencies> ... </dependencies>`:

```
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-csv</artifactId>
  <version>1.10.0</version>
</dependency>
```

> By default Maven belum menambahkan block `dependencies` pada file `pom.xml`, jika memang demikian kita perlu menambahakn blok tersebut terlebih dahulu.

Modifikasi Class `PasswordStore`

Class `PasswordStore` yang telah kita buat sebelumnya memerlukan beberapa penambahan method agar dapat digunakan dengan baik. Berikut adalah versi final setelah perubahan pada class tersebut:

```
public class PasswordStore {
    public String name, username;
    private String password, hashkey;
    private double score;
    private int category;

    public static final int UNCATEGORIZED = 0;
    public static final int CAT_WEBAPP = 1;
    public static final int CAT_MOBILEAPP = 2;
    public static final int CAT_OTHER = 3;

    public static final String [] CATEGORIES = {"Belum terkategori",
                                    "Aplikasi Web", "Aplikasi Mobile", "Akun Lainnya"};

    public PasswordStore(String name, String username, String plainPass){
        this(name, username, plainPass, UNCATEGORIZED);
    }

    public PasswordStore(String name, String username, String plainPass, int category){
        try {
            this.hashkey = Encryptor.generateKey();
        } catch (NoSuchAlgorithmException ex) {
```

```
            Logger.getLogger(PasswordStore.class.getName()).log(Level.SEVERE, null, ex);
        }

        this.name = name;
        this.username = username;

        this.setPassword(plainPass);
        this.setCategory(category);
    }

    public PasswordStore(String name, String username, String encPass,
                                            int category, String hashKey, double score){
        this.name = name;
        this.username = username;
        this.password = encPass;
        this.category = category;
        this.hashkey = hashKey;
        this.score = score;
    }

    public void setEncryptedPass(String encryptedPass, String hashkey){
        this.password = encryptedPass;
        this.hashkey = hashkey;
    }

    public void setPassword(String plainPass){
        String encryptedPass;
        try {
            encryptedPass = Encryptor.encrypt(plainPass, this.hashkey);
            this.password = encryptedPass;
            this.calculateScore(plainPass);
        } catch (Exception ex) {
            Logger.getLogger(PasswordStore.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public String getEncPassword(){
        return this.password;
    }

    public String getHashkey(){
        return this.hashkey;
    }

    public double getScore(){
        return this.score;
    }

    public int getCategoryCode(){
        return this.category;
    }

    public String getPassword(){
        try {
            return Encryptor.decrypt(this.password, this.hashkey);
        } catch (Exception ex) {
            Logger.getLogger(PasswordStore.class.getName()).log(Level.SEVERE, null, ex);
        }
        return null;
    }
```

```
    public void setCategory(int category){
        if(category >= 0 || category <= 3){
            this.category = category;
        } else {
            this.category = 0;
        }
    }

    public String getCategory(){
        return CATEGORIES[this.category];
    }

    private void calculateScore(String plainPass){
        double len = plainPass.length();
        if(len > 15){
            this.score = 10;
        } else {
            this.score = (len / 15) * 10;
        }
    }

    @Override
    public String toString() {
        return this.username+" - "+this.password+" - "+this.hashkey+" - "
                        + String.format("%,.2f", this.score);
    }
}
```

Operasi Baca Tulis CSV

Sebelumnya, kita sudah memiliki class `DataPassword` untuk menyimpan variable static seperti berikut:

```
public class DataPassword {
    public static final ArrayList<PasswordStore> passData = new ArrayList<>();
}
```

Sekarang, kita perlu menambahkan 2 atribut static lain untuk properti penyimpanan file CSV-nya. Berikut adalah atribut yang perlu ditambahkan:

```
private static final String csvPath = "./datapassword.csv";
private static final String [] headers = {"name", "username", "password",
                        "hashkey", "category", "score"};
```

Selanjutnya kita perlu menambahkan 2 method static yaitu `saveCSVData()` untuk menyimpan data CSV dan `loadCSVData()` untuk membaca file CSV.

Berikut adalah koding dari method penyimpanannya:

```
public static void saveCSVData(){
   if(passData.isEmpty()){
      System.out.println("data masih kosong");
   } else {
      try {
         FileWriter writer = new FileWriter("./datapassword.csv");
         CSVFormat formater = CSVFormat.DEFAULT.builder().setHeader(headers).build();
```

```
            CSVPrinter printer = new CSVPrinter(writer, formater);
            for(PasswordStore pass: passData){
               printer.printRecord(pass.name, pass.username, pass.getEncPassword(),
                     pass.getHashkey(), pass.getCategoryCode(), pass.getScore());
            }
            printer.flush();
         } catch (IOException ex) {
            Logger.getLogger(DataPassword.class.getName()).log(Level.SEVERE, null, ex);
         }
      }
   }
```

Sedangkan berikut adalah koding dari method pembacaan file CSV:

```
public static ArrayList<PasswordStore> loadCSVData(){
   passData.clear();
   try {
      FileReader reader = new FileReader(csvPath);
      CSVFormat format = CSVFormat.DEFAULT.builder().setHeader(headers)
                           .setSkipHeaderRecord(true).build();

      Iterable<CSVRecord> data = format.parse(reader);
      for(CSVRecord record: data){
         PasswordStore newPass;
         if(record.get("hashkey") == null){
             newPass = new PasswordStore(
                           record.get("name"),
                           record.get("username"),
                           record.get("password"),
                           Integer.parseInt(record.get("category")));
         } else {
            newPass = new PasswordStore(
                           record.get("name"),
                           record.get("username"),
                           record.get("password"),
                           Integer.parseInt(record.get("category")),
                           record.get("hashkey"),
                           Double.parseDouble(record.get("score")));
         }
         passData.add(newPass);
      }
   } catch (FileNotFoundException ex) {
      System.out.println("Data password kosong ...");
   } catch (IOException ex) {
      Logger.getLogger(DataPassword.class.getName()).log(Level.SEVERE, null, ex);
   }
   return passData;
}
```

Implementasikan Dalam Project

Setelah memiliki method untuk membaca file CSV serta menyimpan data pada file CSV, sekarang kita harus memanggilnya pada project dengan ketentuan sebagai berikut:

- Saat pertama kali dijalankan, program akan memanggil method untuk pembacaan file CSV terlebih dahulu, serta menyimpan datanya pada variabel static `passData`.
  - Jika file CSV tidak ditemukan, berarti variable `passData` masih kosong.

- Semua operasi input data password serta menampilkan data password selama program berjalan, dilakukan pada variabel static `passData` tersebut.
- Sebelum aplikasi ditutup (menu **Keluar** dipilih), maka program akan menulis semua data password yang tersimpan pada variabel `passData` ke dalam file CSV dengan ketentuan hanya encrypted text dan hashkey-nya saja yang tersimpan.