STUDY HARD. DO GOOD AND THE GOOD LIFE WILL FOLLOW.

Putra Pandu Adikara, S.Kom., M.Kom.
2025

Tugas Studi Kasus 1 – Pemrograman Dasar

A. ATURAN PENGERJAAN

- Baca dengan baik-baik dan teliti, jangan sampai ada kesalahan, baik itu format penulisan, kerapian, dll yang dapat mengakibatkan pengurangan nilai.
- Dikerjakan secara berkelompok **maksimal 5 (lima) orang, kalau jumlah kelompok ganjil maka 1-2 kelompok terakhir bisa berjumlah 4 (empat) orang (mayoritas yang 5 orang).**
- **Semua anggota mengerjakan sendiri-sendiri dan mengumpulkan**, namun pilih hanya satu yang digunakan untuk dinilai secara kolektif sebagai nilai kelompok. Mahasiswa yang sudah bisa turut membantu yang belum dalam satu kelompok (**tidak boleh membantu kelompok lain).**
- Dikerjakan dengan bahasa pemrograman Java, menggunakan tools Netbeans/Eclipse.

B. ATURAN PLAGIARISME

- **Dilarang menjiplak/menyontek** dengan alasan, cara, sesedikit apapun.
- **Setiap kode program akan dicek** secara otomatis menggunakan program untuk mengetahui **tingkat kemiripan (plagiarisme).** Nama variabel diubah atau letaknya dipindah tetap dihitung sama.
- **Tingkat kemiripan di atas 50%** dari program akan diberi nilai 0 setelah melalui pertimbangan manual.
- **Dilarang menggunakan tools AI (ChatGPT),** meski sebagus apa pun nilai Anda pada akhirnya akan percuma kalau logika berpikir tidak berjalan, Anda akan kesulitan ketika PKL/Magang, Skripsi, atau bahkan bekerja nanti. Tujuan tugas ini salah satunya untuk membentuk ***soft skill,*** khususnya kemampuan ***computational thinking, critical thinking***. Percuma kalau AI yang cerdas tapi manusia kurang cerdas.

C. PENGUMPULAN TUGAS

- Batas waktu pengumpulan di BRONE, **perhatikan jam *server***.
- ***Soft copy* melalui ke BRONE, lampirkan SEMUA FAIL (**.DOCX/PDF, .JAVA, dll.) **dengan terkompres ZIP/RAR** dengan format:

  `[TX][PD-PRODI-KELAS] NamaMahasiswa.ZIP`

  - **TX** adalah kode tugas, apabila Tugas 1 maka **T1**, Tugas 2 maka **T2**, dst. Tugas Akhir/Proyek Akhir **TP**
  - **PRODI** diisi inisial prodi: **TIF** (Teknik Informatika), **TKOM** (Teknik Komputer), **SI** (Sistem Informasi)
  - **KELAS** diisi kode kelas misal **A**, **B**, C**,** dst.
  - **NamaMahasiswa** maksudnya tidak perlu menuliskan semua nama anggota, tapi nama satu anggota saja, sebagai ketua kelompok/penanggung jawab unggahan.

  Contoh: `[TP][PD-TIF-B] Elon Gates.ZIP` → Tugas 2, kelas PD TIF-B

  Di dalam fail .ZIP di atas berisi:
  - Fail **.DOCX/.PDF** untuk **Flowchart, Pseudocode, Source Code** (seluruh jawaban soal jadi satu dokumen, lihat bagian Format dan Struktur)
  - Fail **.JAVA** dengan format: `TP_NIM_Nama_NoSoal.java` (semua class dalam satu file saja)
    - Misalnya:
      `TP_123456789_Nama_1.java` (tentunya nama ***public class*** disamakan dengan nama fail)
  - Fail lainnya yang diperlukan

D. Format dan Struktur Pengerjaan Tugas

Dalam file .DOCX/PDF berisi contoh struktur sbb.:

1. Soal (tidak perlu ditulis, cukup nomornya saja)
   a. ***Pseudocode*** (diketik dengan *font* Cambria/Times New Roman seperti contoh berikut)

	Algoritme untuk Menampilkan Input dan Ouput *{ Algoritme ini digunakan untuk menampilkan input dan output }*
1	**DEKLARASI**
2	x: integer
3	**DESKRIPSI**
4	Cetak "Masukkan x"
5	Input x
6	**if** x % 2 == 0 **then**
7	x ← x+1
8	**else**
9	x ← x+2
10	Cetak "Hasilnya adalah ", x
11	**end if**

   b. ***Flowchart***

- Gambar flowchart di sini –

   c. ***Source Code*** *(*diketik dengan *font* Consolas/Courier New seperti contoh berikut)

Source Code Untuk Menampilkan Input dan Ouput

```
1   /**
2   Nama 1: NAMA MAHASISWA 1 (NIM) *)
3   Nama 2: NAMA MAHASISWA 2 (NIM)
4   ...
    *) Tugas yang dinilaikan secara kolektif

5   Tanggung jawab 1: mengerjakan soal nomor 1, membuat xxx, dll.
6   Tanggung jawab 2: mengerjakan soal nomor 2, membuat xxx, dll.
7
8   **/
9   public class Contoh {
10     public static void main(String[] args) {
11        Scanner input = new Scanner(System.in);
12        System.out.println("Masukkan x");
13        int x = input.nextInt();
14        if (x % 2 == 0) {
15           x = x+1;
16        } else {
17           x = x+2;
18        }
19        System.out.println("Hasilnya adalah " + x);
20     }
21  }
```

   d. **Pembahasan/Penjelasan dari Source Code**

   Contoh pembahasan

   Baris 11, membuat objek *input* dari class Scanner yang berfungsi sebagai masukan dari keyboard. Baris 13, variabel *x* bertipe integer akan diisikan dari masukan keyboard. Baris 14 hingga 15, akan menyeleksi kondisi sesuai isi dari x. Apabila x bernilai genap, maka akan mencetak hasil x+1, sedangkan bila tidak maka akan mencetak x+2.

   e. ***Screenshot***

- Gambar screenshot output (bukan source code) -

2. dst. seperti no. 1

E. TUJUAN TUGAS

Pada tugas ini mahasiswa akan diberikan studi kasus untuk menyelesaikan masalah dalam bentuk esai. Dengan berbekal pengetahuan pemrograman dasar mengenai seleksi kondisi, mahasiswa diharapkan mampu memahami hingga mengimplementasikan struktur kontrol yang tepat untuk digunakan dalam menyelesaikan masalah yang diberikan yang berfokus ke operasi input, *output*, seleksi kondisi, pengulangan dan struktur data *array*. Tidak hanya *hard skill*, tapi tujuan tugas ini juga untuk membentuk *soft skill*, khususnya kemampuan *computational thinking, critical thinking*.

F. DESKRIPSI TUGAS

FILKOMAntar

Anda diminta untuk membuat program simulasi oleh suatu restoran. Program simulasi digunakan untuk memesan makanan hingga mengantarkan makanan dari suatu restoran ke pembeli. Pembeli akan memiliki saldo yang dapat digunakan saat *check-out* pesanan. Apabila saldo kurang maka pembeli dapat melakukan *topup*. Selesai melakukan pemesanan dan *check-out* dilakukan maka program akan mencarikan *driver* yang paling dekat dengan pembeli. Rumus menghitung jarak pembeli dengan *driver* menggunakan rumus jarak Euclidean:

$$dist(pembeli, driver) = \sqrt{(pembeli_x - driver_x)^2 + \left(pembeli_y - driver_y\right)^2}$$

Untuk memastikan keadilan dan kesejahteraan *driver*, maka *driver* yang sudah pernah mendapat pesanan dari pembeli yang sama tidak boleh menerima pesanan lagi dan harus mencari *driver* terdekat lainnya. Dalam kasus ini hanya ada satu pembeli tapi bisa ada banyak *driver* yang dapat ditambahkan sewaktu-waktu. Menu makanan/minuman juga dapat ditambahkan sewaktu-waktu, termasuk pesanan yang dilakukan oleh pembeli.

Struktur data yang digunakan untuk menyimpan data dan *skeleton* program telah disediakan dan WAJIB untuk digunakan.

CATATAN

- **Dilarang mengubah format masukan dan keluaran meskipun hanya satu spasi atau karakter lain (apalagi benar-benar berbeda sesuai contoh) atau dapat berakibat nilai 0. Output dari program akan dicocokkan dengan 10 test case, setiap test case benar bernilai 10. Perhatikan baik-baik format masukan dan keluaran.**
- **Tidak diperbolehkan adanya keluaran apapun saat memberi masukan, mengubah format masukan, dan format keluaran. Jadi bukan input 1 baris keluar 1 output, tapi langsung semua input baru keluar semua output.**
- **JADI, TIDAK SELANG-SELING INPUT-OUTPUT-INPUT-OUTPUT-dst TAPI MEMBACA SEMUA INPUT DULU (MISAL COPY-PASTE) BARU KEMUDIAN SEMUA OUTPUT. INPUT BERAKHIR KETIKA DITEMUKAN BARIS KOSONG atau END OF FILE.**
- **Pengecekan program dilakukan dengan copy-paste input seperti HackerRank.**

Versi 1.1 – apabila ada perubahan akan diberitahukan secepatnya khususnya apabila ada format masukan dan keluaran baru atau revisi. Kerjakan dulu secepatnya karena bisa jadi masih ada kesalahan/revisi dari dokumentasi ini.

Gunakan hanya struktur data berupa array biasa, tanpa membuat *class* dan *object* dan **jangan gunakan** Collections API di Java (ArrayList, Vector, Map, dll).

Format Masukan dan Keluaran

1. **PERINTAH `UPSERT_CUST`**
   - **Deskripsi:** Menampilkan informasi nama dan lokasi pembeli atau memperbarui lokasi pembeli jika sudah ada.
   - **Input:** `UPSERT_CUST NAMA_PEMBELI X Y`
     - `X` adalah koordinat X
     - `Y` adalah koordinat Y

     Contoh: `ANI 25.6 75.1`
   - **Output:** Menampilkan nama dan lokasi pembeli, jika pembeli belum ada dalam daftar
     `CUSTOMER NAMA_PEMBELI DITAMBAH @ x, y`
     Jika pembeli sudah ada, pembeli tidak ditambahkan tapi lokasi pembeli diperbarui
     `CUSTOMER NAMA_PEMBELI SUDAH ADA, LOKASI BARU @ x, y`
2. **PERINTAH `DEL_CUST`**
   - **Deskripsi:** Menghapus informasi pembeli.
   - **Input:** `DEL_CUST NAMA_PEMBELI`
   - **Output:** Jika pembeli ditemukan dalam daftar maka akan dihapus dan tampilkan
     `CUSTOMER: NAMA_PEMBELI BERHASIL DIHAPUS`
     Jika pembeli tidak ditemukan maka tampilkan
     `CUSTOMER: NAMA_PEMBELI TIDAK DITEMUKAN`
3. **PERINTAH `PRINT_CUST`**
   - **Deskripsi:** Menampilkan informasi nama dan lokasi pembeli yang diurutkan berdasarkan nama.
   - **Input:** `PRINT_CUST`
   - **Output:** Menampilkan nama dan lokasi pembeli dalam format
     `NAMA_PEMBELI_1 @ x, y`
     `NAMA_PEMBELI_2 @ x, y`
     `NAMA_PEMBELI_3 @ x, y`
4. **PERINTAH: `UPSERT_DRIVER`**
   - **Deskripsi:** Menambahkan driver baru ke daftar driver atau memperbarui lokasi driver jika sudah ada.
   - **Input:** `UPSERT_DRIVER NAME ID X Y`
     - `NAME`: Nama driver.
     - `ID`: ID driver (nomor plat) misalnya N_5501_JKV
     - `X, Y`: Koordinat lokasi driver.
   - **Output:** Menampilkan nama dan lokasi driver dalam format, jika driver belum ada dalam daftar
     `DRIVER NAMA_DRIVER DITAMBAH @ x, y`
     Jika driver sudah ada, driver tidak ditambahkan tapi lokasi driver diperbarui
     `DRIVER NAMA_DRIVER SUDAH ADA, LOKASI BARU @ x, y`

5. PERINTAH `DEL_DRIVER`

- **Deskripsi:** Menghapus informasi driver.
- **Input:** `DEL_DRIVER NAMA_DRIVER`
- **Output:** Jika pembeli ditemukan dalam daftar maka akan dihapus dan tampilkan
  `DRIVER: NAMA_DRIVER BERHASIL DIHAPUS`
  Jika pembeli tidak ditemukan maka tampilkan
  `DRIVER: NAMA_DRIVER TIDAK DITEMUKAN`

6. PERINTAH: `CALCULATE_DISTANCE`

- **Deskripsi:** Menghitung jarak antara pembeli ke driver dan tampilannya diurutkan berdasarkan **jarak terdekat ke terjauh (*ascending*).**
- **Input:** `CALCULATE_DISTANCE NAMA_PEMBELI`
- **Output:** Menampilkan nama dan lokasi serta jarak dari pembeli ke driver. `aa.bb` merupakan jarak dalam bilangan desimal, dua digit di belakang koma.

```
DISTANCE NAMA_PEMBELI @ x, y
NAMA_DRIVER_1 @ x, y = aa.bb
NAMA_DRIVER_2 @ x, y = aa.bb
NAMA_DRIVER_3 @ x, y = aa.bb
```

7. PERINTAH: `PRINT_DISTANCE_MATRIX`

- **Deskripsi:** Menghitung jarak semua pembeli ke semua driver berdasarkan urutan saat dimasukkan. Dalam program, buat matriks ini dalam bentuk array multidimensi.
- **Input:** `PRINT_DISTANCE_MATRIX`
- **Output:** Tabel/matriks berisi jarak dari semua pembeli ke semua driver berdasarkan urutan masukan. `aa.bb` merupakan jarak dalam bilangan desimal, dua digit di belakang koma. Misalnya untuk pembeli bernama Budi dimasukkan pertama kemudian Ani dan driver bernama Zaza, dan Yaya, maka contoh hasilnya:

```
[Budi][Zaza][aa.bb]
[Budi][Yaya][aa.bb]
[Ani][Zaza][aa.bb]
[Ani][Yaya][aa.bb]
```

8. PERINTAH: `CALCULATE_COST`

- **Deskripsi:** Menghitung biaya antara pembeli ke semua driver dan tampilannya diurutkan berdasarkan **jarak terdekat ke terjauh (*ascending*).**
- **Input:** `CALCULATE_COST NAMA_PEMBELI`
- **Output:** Menampilkan nama dan lokasi serta jarak dari pembeli ke driver. `aaaaa` merupakan biaya dalam bilangan bulat.

```
DISTANCE NAMA_PEMBELI @ x, y
NAMA_DRIVER_1 @ x, y = aaaaa
NAMA_DRIVER_2 @ x, y = aaaaa
NAMA_DRIVER_3 @ x, y = aaaaa
```

Rumus yang digunakan untuk perhitungan biaya:
Jika < 1 km (1 km pertama) maka biaya dasar = 7000
Jika 1 < jarak <= 5 km maka biaya dasar ditambah 2000 setiap km
Jika 5 < jarak <= 10 km maka biaya dasar ditambah 2500 setiap km
Jika > 10 km maka biaya dasar ditambah 3000 setiap km
Gunakan pembulatan ke atas.

Contoh:
Jarak: 0,5 maka biaya 7000
Jarak: 1,0 maka biaya 7000
Jarak: 1,3 maka biaya 7000 + 2000 = 9000
Jarak: 1,8 maka biaya 7000 + 2000 = 9000
Jarak: 2,8 maka biaya 7000 + 4000 = 11000

FLOWCHART

Buat flowchart untuk proses `UPSERT_CUST` dan `CALCULATE_COST`. Tidak perlu membuat flowchart untuk semua bagian program secara utuh.

CONTOH INPUT

Catatan: urutan perintah bisa berbeda-beda dan akan diuji dengan input test case yang berbeda-beda tidak hanya yang ada di contoh ini.

```
UPSERT_CUST Budi 10.5 6.7
UPSERT_CUST Ani 9.3 8.5
UPSERT_CUST Caca 3.5 10.6
PRINT_CUST
UPSERT_CUST Bela 2.7 3.4
UPSERT_CUST Caca 5.5 7.6
PRINT_CUST
DEL_CUST Budi
PRINT_CUST
UPSERT_DRIVER Zaza 5.6 7.6
UPSERT_DRIVER Yaya 10.6 9.1
UPSERT_DRIVER Wili 7.9 4.3
CALCULATE_DISTANCE Ani
CALCULATE_DISTANCE Caca
PRINT_DISTANCE_MATRIX
```

CONTOH OUTPUT

```
CUSTOMER: Budi DITAMBAH @ 10.5, 6.7
CUSTOMER: Ani DITAMBAH @ 9.3, 8.5
CUSTOMER: Caca DITAMBAH @ 3.5, 10.6
Ani @ 9.3, 8.5
Budi @ 10.5, 6.7
Caca @ 3.5, 10.6
CUSTOMER: Bela DITAMBAH @ 2.7, 3.4
CUSTOMER: Caca SUDAH ADA, LOKASI BARU @ 5.5, 7.6
Ani @ 9.3, 8.5
Bela @ 2.7, 3.4
Budi @ 10.5, 6.7
Caca @ 5.5, 7.6
CUSTOMER: Budi BERHASIL DIHAPUS
Ani @ 9.3, 8.5
Bela @ 2.7, 3.4
Caca @ 5.5, 7.6
DRIVER: Zaza DITAMBAH @ 5.6, 7.6
DRIVER: Yaya DITAMBAH @ 10.6, 9.1
DRIVER: Wili DITAMBAH @ 7.9, 4.3
DISTANCE Ani @ 9.3, 8.5
```

```
Yaya @ 10.6, 9.1 = 1.43
Zaza @ 5.6, 7.6 = 3.81
Wili @ 7.9, 4.3 = 4.43
DISTANCE Caca @ 5.5, 7.6
Zaza @ 5.6, 7.6 = 0.10
Wili @ 7.9, 4.3 = 4.08
Yaya @ 10.6, 9.1 = 5.32
[Ani][Zaza][3.81]
[Ani][Yaya][1.43]
[Ani][Wili][4.43]
[Caca][Zaza][0.10]
[Caca][Yaya][5.32]
[Caca][Wili][4.08]
[Bela][Zaza][5.10]
[Bela][Yaya][9.74]
[Bela][Wili][5.28]
```