Soal 1

Kelas **HideNSeek** merupakan sebuah kelas yang memodelkan permainan Petak Umpet

Buatlah sebuah kelas **HideNSeek** yang memiliki atribut:

1. **seeker** dengan tipe data **String**, digunakan untuk menyimpan nama seeker.
2. **player** dengan tipe data **int**, digunakan untuk menyimpan banyak pemain selain seeker.
3. **playerFound** dengan tipe data **int**, digunakan untuk menyimpan jumlah pemain yang telah ditemukan oleh seeker.

Lengkapi dan submit file **HideNSeek.java**

HideNSeek.java

```
public class HideNSeek {
    private String seeker;
    private int player;
    private int playerFound;

    /**
    * HideNSeek constructor
    * @param seeker
    * @param player
    * /
    */
    public HideNSeek(String seeker, int player) {

    }

    /**
     * getSeeker
     * @return seeker
     */
    public String getSeeker() {

    }

    /**
     * getPlayer
     * @return player
     */
    public int getPlayer() {
```

```
    }

    /**
     * getPlayerFound
     * @return playerFound
     */
    public int getPlayerFound() {

    }

    /**
     * Menambahkan playerFound sebanyak 1
     * dengan syarat masih ada player yang belum ditemukan
     * @return void
     */
    public void foundPlayer() {

    }
}
```

Soal 2

Kelas **CarDealer** merupakan sebuah kelas yang memodelkan dealer mobil

Buatlah sebuah kelas **CarDealer** yang memiliki atribut:

1. **brand** dengan tipe data **String**, digunakan untuk menyimpan nama brand mobil.
2. **carInDealer** dengan tipe data **int**, digunakan untuk menyimpan banyak mobil yang ada di dealer.
3. **carPrice** dengan tipe data **int**, digunakan untuk menyimpan harga mobil.
4. **profit** dengan tipe data **int**, digunakan untuk menyimpan total uang yang didapatkan dari menjual mobil.
5. **totalCar** dengan tipe data **int** dan merupakan static variable, digunakan untuk menyimpan total mobil yang ada di semua dealer.

Lengkapi dan submit file **CarDealer.java**

CarDealer.java

```
public class CarDealer {
    private static int totalCar = 0;
    private String brand;
    private int carInDealer;
    private int carPrice;
    private int profit = 0;

    /**
     * CarDealer constructor
     * Inisiasi sekaligus menambah totalCar
     * @param brand
     * @param carInDealer
     * @param carPrice
     */
    public CarDealer(String brand, int carInDealer, int carPrice) {

    }

    /**
     * getBrand
     * @return brand
     */
    public String getBrand() {
```

```
    }

    /**
     * getCarInDealer
     * @return carInDealer
     */
    public int getCarInDealer() {

    }

    /**
     * getCarPrice
     * @return carPrice
     */
    public int getCarPrice() {

    }

    /**
     * getProfit
     * @return profit
     */
    public int getProfit() {

    }

    /**
     * getTotalCar
     * @return totalCar
     */
    public static int getTotalCar() {

    }

    /**
     * sellCar
     * Menjual mobil sebanyak amount. Ketika berhasil, mengurangi mobil di
dealer dan juga totalCar
     * sekaligus menambah profit sesuai dengan harga mobil dan jumlah mobil
yang terjual
     * Prekondisi: Mobil mungkin tidak cukup
     * @param amount
     * @return void
     */
    public void sellCar(int amount) {

    }
}
```

Soal 3

Ibnu ingin bermain petak umpet bersama teman-temannya. Namun, karena di luar sedang terjadi tornado, ia dan teman-temannya terpaksa bermain petak umpet secara virtual.

Buatlah program untuk Ibnu dan teman-temannya bermain petak umpet virtual. Beri program dengan nama file **GameSimulator.java** dan gunakan kelas/file **HideNSeek.java** yang sudah dibuat pada soal nomor 1 untuk mengerjakan soal berikut.

Program dimulai dengan memasukkan jumlah pemain, kemudian memasukkan siapa yang menjadi seeker. Selanjutnya program akan meminta input terus-menerus hingga semua pemain telah ditemukan. Input untuk membaca pemain ditemukan adalah jumlah permain ditemukan saat ini + 1, jika program mendapatkan input selain itu, maka program akan menganggap pemain lain belum ditemukan.

Format Input:

- Baris pertama merupakan jumlah pemain petak umpet
- Baris kedua merupakan pemain yang menjadi seeker
- N baris selanjutnya adalah angka yang mengindikasikan apakah pemain ditemukan atau tidak.

Jika pemain ditemukan input harus x + 1. Dimana x adalah jumlah pemain yang ditemukan saat ini. Selain input di atas, program akan menganggap pemain belum ditemukan. Program akan meminta input lagi.

Format Output:

- Setelah menerima input seeker. Program akan mengeluarkan output "Game dimulai dengan {jumlah pemain} pemain, seeker adalah {seeker}"
- Jika seeker menemukan pemain maka program akan mengeluarkan output "{n} Pemain ditemukan". Dimana n adalah jumlah pemain yang ditemukan
- Jika seeker telah menemukan semua pemain. Program akan mengeluarkan output "Semua pemain telah ditemukan, permainan selesai."

Contoh:

Input	Output	Penjelasan
5 Ibnu 5 1 2 4 3 4	Game dimulai dengan 5 pemain, seeker adalah Ibnu 1 Pemain ditemukan 2 Pemain ditemukan 3 Pemain ditemukan 4 Pemain ditemukan Semua pemain telah ditemukan, permainan selesai.	Game dimulai Pemain ditemukan 0 Pemain ditemukan 1 Pemain ditemukan 2 Pemain ditemukan 2 Pemain ditemukan 3 Pemain ditemukan 4

Note:

- Permainan pasti dilakukan dengan minimal 2 orang
- Semua output diakhiri dengan newline
- Gunakan **scanner.nextLine()** untuk menerima input string yang diakhiri newline dan **Integer.parseInt(scanner.nextLine())** untuk menerima input integer yang diakhiri newline