NETWORK LABORATORY
Electrical Engineering Department, 2nd floor
Universitas Indonesia
Depok. 16424

PEMROGRAMAN BERORIENTASI OBJEK 2017/2018

CASE STUDY

MODUL 6: Array dan ArrayList

Untuk melanjutkan program JHotel yang dibuat, harap untuk melihat UML Modul 6 yang telah disediakan. **Praktikan diharapkan menyesuaikan program JHotel sesuai dengan UML Modul 6!**

Pada modul ini, sistem JHotel yang masih bersifat **single-user** akan diubah menjadi **multi-user**, dengan mengubah implementasi seluruh database Class.

Pada modul ini data yang dikumpulkan adalah **2 commit** pada GitHub:

- **Commit 1: Tugas 1-7**, dilakukan sebelum waktu praktikum berakhir
- **Commit 2: Tugas 8-12**, dilakukan sebelum presentasi dengan asisten dilakukan

Hint: gunakan for-each loop setiap kali melakukan akses kepada suatu array atau Collection (List, ArrayList, dll) untuk mempercepat pembuatan program.

Tugas 1: Pemindahan IDE

1. Backup seluruh source code yang telah dibuat pada modul-modul sebelumnya.
2. Kosongkan folder project BlueJ sebelumnya dengan menghapus seluruh source code yang ada **(FOLDER .git JANGAN DIHAPUS)**
3. Commit perubahan folder yang telah dilakukan agar isi repository pada GitHub juga menjadi kosong.
4. Buatlah sebuah Java Project baru pada IDE pilihan Anda. Pilih direktori project ke folder project BlueJ sebelumnya dikosongkan. Pindahkan seluruh code (class dan enum) yang telah di-backup sebelumnya kedalam folder ”src” yang ada pada IntelliJ Project yang telah dibuat.

Tugas 2: Penyesuaian UML

1. Ubah constructor dari class `Pesanan` agar tidak menerima parameter object `Room` (pertama kali pesanan dibuat, diinginkan agar tidak memiliki kamar pesanan terlebih dahulu).
2. Pada class `Room`, lakukan penghapusan variable `isAvailable` dan accessor dan mutator method untuk variable tersebut, serta hilangkan `isAvailable` dari constructor class tersebut.

Tugas 3: Class `Pesanan` pt. 1

1. Tambahkan variable `id` beserta accessor dan mutator-nya sesuai dengan UML Modul 6.
2. Tambahkan variable `isAktif` beserta accessor dan mutator-nya sesuai dengan UML Modul 6.
3. Pada constructor class, buatlah agar setiap object `Pesanan` yang dibuat akan memiliki nilai `isAktif` awal `true` dan `tanggalPesan` akan berisi tanggal ketika constructor tersebut dipanggil.
4. Pada class `Administrasi`, ketika setiap pesanan diselesaikan atau dibatalkan (dengan method `pesananDibatalkan()` atau `pesananSelesai()` dengan parameter apapun), buatlah agar status aktif dari pesanan tersebut menjadi `false`.

Tugas 4: Class `DatabasePesanan`

1. Buatlah variable dan method sesuai dengan UML Modul 6.
2. Instantiate variable `PESANAN_DATABASE` dan berikan nilai 0 untuk variable `LAST_PESANAN_ID`.
3. Method `getPesananDatabase()` dan `getLastPesananID ()` merupakan accessor untuk masing-masing variable.
4. Implementasikan method `addPesanan()` dengan ketentuan sebagai berikut:
   - Pesanan bisa ditambahkan kedalam database jika pelanggan yang membuat pesanan tersebut tidak memiliki pesanan dengan status aktif lainnya didalam database (1 pelanggan hanya bisa memiliki 1 pesanan dengan status aktif didalam database)
   - Jika kondisi terpenuhi, tambahkan pesanan kedalam database, lalu kembalikan nilai `true` pertanda penambahan berhasil dilakukan
   - Jika kondisi tidak terpenuhi, jangan tambahkan pesanan tersebut kedalam database, dan kembalikan nilai `false` pertanda penambahan gagal dilakukan
5. Implementasikan method `getPesanan(int id)` dengan ketentuan sebagai berikut:

- Pada database, carilah pesanan yang memiliki ID sesuai dengan parameter method
- Jika ditemukan, kembalikan pesanan tersebut
- Jika tidak ditemukan, kembalikan nilai `null`

6. Implementasikan method `getPesanan(Room kamar)` dengan ketentuan sebagai berikut:
   - Pada database, carilah pesanan yang dilakukan untuk kamar pada parameter method
   - Jika ditemukan, kembalikan pesanan tersebut
   - Jika tidak ditemukan, kembalikan nilai `null`
7. Implementasikan method `getPesananAktif()` dengan ketentuan sebagai berikut:
   - Pada database, carilah pesanan yang dibuat oleh pelanggan pada parameter method dan masih berstatus aktif
   - Jika ditemukan, kembalikan pesanan tersebut
   - Jika tidak ditemukan, kembalikan nilai `null`
8. Implementasikan method `removePesanan()` dengan ketentuan sebagai berikut:
   - Pada database, carilah pesanan yang dibuat oleh pelanggan pada parameter method
   - Jika pesanan tersebut masih memiliki kamar, maka batalkan pesanan tersebut telebih dahulu.
   - Jika pesanan tersebut tidak memiliki kamar, namun masih bersatus aktif, maka ubah status aktif dari pesan tersebut menjadi `false`
   - Setelah kedua pengecekan diatas dilakukan, hapus pesanan tersebut dari database dan kembalikan nilai `true` pertanda penghapusan berhasil dilakukan
   - Jika pesanan tidak ditemukan, kembalikan nilai `false` pertanda penghapusan gagal dilakukan

Tugas 5: Class `Pesanan` pt. 2

1. Pada constructor class `Pesanan`, nilai `id` akan di-assign sesuai dengan nilai ID pesanan terakhir yang terdaftar pada database ditambah dengan 1.

Tugas 6: Class `DatabaseRoom`

1. Buatlah variable dan method sesuai dengan UML Modul 6.
2. Instantiate variable `ROOM_DATABASE`.
3. Method `getRoomDatabase()` merupakan accessor untuk variable yang ada.
4. Implementasikan method `addRoom()` dengan ketentuan sebagai berikut:

- Kamar bisa ditambahkan kedalam database jika kamar tersebut memiliki hotel dan nomor kamar yang tidak sama dengan kamar lainnya yang sudah terdaftar didalam database (pada satu hotel, tidak boleh terdapat 2 kamar dengan nomor yang sama)
- Jika kondisi terpenuhi, tambahkan kamar kedalam database, lalu kembalikan nilai `true` pertanda penambahan berhasil dilakukan
- Jika kondisi tidak terpenuhi, jangan tambahkan kamar tersebut kedalam database, dan kembalikan nilai `false` pertanda penambahan gagal dilakukan

5. Implementasikan method `getRoom()` dengan ketentuan sebagai berikut:
   - Pada database, carilah kamar yang dimiliki hotel dan memiliki nomor kamar sesuai dengan parameter method
   - Jika ditemukan, kembalikan kamar tersebut
   - Jika tidak ditemukan, kembalikan nilai `null`
6. Implementasikan method `getRoomsFromHotel()` dengan ketentuan sebagai berikut:
   - Pada database, carilah kamar yang dimiliki hotel sesuai dengan parameter method
   - Kembalikan seluruh list kamar tersebut dalam sebuah `ArrayList`
7. Implementasikan method `getVacantRooms()` dengan ketentuan sebagai berikut:
   - Pada database, carilah kamar yang memiliki status `VACANT`
   - Kembalikan seluruh list kamar tersebut dalam sebuah `ArrayList`
8. Implementasikan method `removeRoom()` dengan ketentuan sebagai berikut:
   - Pada database, carilah kamar dengan nama yang sesuai dengan parameter method
   - Batalkan pesanan dari kamar tersebut terlebih dahulu jika ada, lalu hapus kamar tersebut dan kembalikan nilai `true` pertanda penghapusan berhasil dilakukan
   - Jika kamar tidak ditemukan, kembalikan nilai `false` pertanda penambahan gagal dilakukan

Tugas 7: Class `Room`

1. Pada constructor class `Room`, buatlah agar setiap kali object `Room` dibuat, nilai `statusKamar` awal yang dimiliki object akan selalu "vacant."
2. Hapus variable `pesanan` dari class `Room` beserta accessor dan mutator dari variable tersebut.
3. Pada method `toString()` yang mengharuskan untuk mengetahui apakah kamar memiliki pesanan atau tidak, carilah pesanan yang dilakukan untuk kamar tersebut pada database dengan menggunakan method yang sesuai pada class `DatabasePesanan`.
4. Gantilah seluruh pemanggilan method `Room.getPesanan()` pada seluruh class dengan melakukan pencarian pesanan yang dilakukan untuk kamar yang bersangkutan pada database dengan menggunakan method yang sesuai pada class `DatabasePesanan`.
5. Pada class `Administrasi`, method `roomAmbilPesanan()` dan `roomAmbilPesanan()` tidak perlu lagi memanggil method `Room.setPesanan()` karena class `Room` tidak memiliki object `Pesanan` lagi. Sehingga, method tersebut hanya perlu melakukan pengubahan nilai `status_kamar` pada object `Room` pada parameter method. Hapuslah method `roomAmbilPesanan()` dan ganti setiap pemanggilannya dengan melakukan pengubahan nilai `status_kamar` pada object `Room` menjadi "booked," dan hapuslah method `roomLepasPesanan()` dan ganti setiap pemanggilannya dengan melakukan pengubahan nilai `status_kamar` pada object `Room` menjadi "vacant."

(**Hint nomor 2-5:** Kita dapat melihat seluruh pemanggilan variable/method tersebut pada seluruh class dengan meng-klik kanan pada variable/method tersebut, lalu pilih "Find Usages" atau dengan menekan `Alt+F7`, sehingga seluruh pengubahan implementasi yang perlu dilakukan dapat dilihat)

Tugas 8: Class `Hotel` pt. 1

1. Tambahkan variable baru `id` serta accessor dan mutator untuk variable tersebut sesuai dengan UML Modul 6.

Tugas 9: Class `DatabaseHotel`

1. Buatlah variable dan method sesuai dengan UML Modul 6.
2. Instantiate variable `HOTEL_DATABASE` dan berikan nilai awal 0 untuk variable `LAST_HOTEL_ID`.

3. Method `getHotelDatabase()` dan `getLastHotelID ()` merupakan accessor untuk masing-masing variable.
4. Implementasikan method `addHotel()` dengan ketentuan sebagai berikut:
   a. Hotel bisa ditambahkan kedalam database jika ID hotel tersebut belum dimiliki oleh hotel lainnya yang sudah terdaftar didalam database
   b. Jika kondisi terpenuhi, tambahkan hotel kedalam database, lalu ubah nilai `LAST_HOTEL_ID` menjadi nilai ID dari hotel yang sebelumnya ditambahkan dan kembalikan nilai `true` pertanda penambahan berhasil dilakukan
   c. Jika kondisi tidak terpenuhi, jangan tambahkan hotel tersebut kedalam database, dan kembalikan nilai `false` pertanda penambahan gagal dilakukan
5. Implementasikan method `getHotel()` dengan ketentuan sebagai berikut:
   a. Pada database, carilah hotel dengan ID yang sesuai dengan parameter method
   b. Jika ditemukan, kembalikan hotel tersebut
   c. Jika tidak ditemukan, kembalikan nilai `null`
6. Implementasikan method `removeHotel()` dengan ketentuan sebagai berikut:
   a. Pada database, carilah hotel dengan ID yang sesuai dengan parameter method
   b. Jika hotel ditemukan, hapus seluruh kamar tersebut terlebih dahulu (**gunakan method `getRoomsFromHotel()` dan `removeRoom()`**), lalu hapus hotel tersebut dan kembalikan nilai `true` pertanda penghapusan berhasil dilakukan
   c. Jika hotel tidak ditemukan, kembalikan nilai `false` pertanda penambahan gagal dilakukan

Tugas 10: Class `Hotel` pt. 2

1. Pada constructor class, nilai `id` akan di-assign sesuai dengan nilai ID hotel terakhir yang terdaftar pada database ditambah dengan 1.

Tugas 11: Class `DatabaseCustomer`

1. Buatlah variable dan method sesuai dengan UML Modul 6.
2. Instantiate variable `CUSTOMER_DATABASE` dan berikan nilai awal 0 pada variable `LAST_CUSTOMER_ID`.

3. Method `getCustomerDatabase()` dan `getLastCustomerID()` merupakan accessor untuk masing-masing variable.
4. Implementasikan method `addCustomer()` dengan ketentuan sebagai berikut:
   - Pelanggan bisa ditambahkan kedalam database jika belum ada pelanggan dengan nilai ID yang sama pada database
   - Jika kondisi terpenuhi, tambahkan pelanggan kedalam database, lalu ubah nilai `LAST_CUSTOMER_ID` dengan ID dari pelanggan yang sebelumnya ditambahkan dan kembalikan nilai `true` pertanda penambahan berhasil dilakukan
   - Jika kondisi tidak terpenuhi, jangan tambahkan pelanggan tersebut kedalam database, dan kembalikan nilai `false` pertanda penambahan gagal dilakukan
5. Implementasikan method `getCustomer()` dengan ketentuan sebagai berikut:
   - Pada database, carilah pelanggan yang memiliki nilai ID yang sama dengan parameter method
   - Jika ditemukan, kembalikan pelanggan tersebut
   - Jika tidak ditemukan, kembalikan nilai `null`
6. Implementasikan method `removeCustomer()` dengan ketentuan sebagai berikut:
   - Pada database, carilah pelanggan yang memiliki nilai ID yang sama dengan parameter method
   - Jika pelanggan ditemukan, hapus pesanan oleh pelanggan tersebut terlebih dahulu (**gunakan method dan `removePesanan()`**), lalu hapus pelanggan tersebut dan kembalikan nilai `true` pertanda penghapusan berhasil dilakukan
   - Jika pelanggan tidak ditemukan, kembalikan nilai `false` pertanda penghapusan gagal dilakukan

Tugas 12: Class `Customer`

1. ID dari setiap pelanggan akan dibuat menjadi nilai incremental dari 1 hingga seterusnya. Hapus parameter `id` dari kedua constructor class `Customer`. Nilai `id` tidak dimasukkan pada parameter, melainkan diambil dari nilai `LAST_CUSTOMER_ID` ditambahkan dengan 1.
2. Pada method `toString()` yang telah dibuat pada Modul 5 sebelumnya, terdapat perbedaan nilai output tergantung pada apakah pelanggan tersebut memiliki pesanan aktif atau tidak. Lakukan pencarian pesanan aktif oleh pelanggan tersebut pada class `DatabasePesanan` dengan menggunakan method yang telah dibuat.

NETWORK LABORATORY
Electrical Engineering Department, 2nd floor
Universitas Indonesia
Depok. 16424

"Try not. Do or do not. There is no try"

Yoda