CSE 1322L – Assignment 2 (Spring 2025)

Introduction

In this assignment, you will write a program that allows a landlord to keep track of their tenants. This landlord can have multiple buildings, each with a fixed number of apartments, each of which will only officially hold a single tenant. The system must also keep track of each tenant's name, their age, and their rent amount. Finally, the landlord's balance must also be kept, and the system must provide ways to update it due to either charging tenants their rent, or because some sort of maintenance needed to be performed.

Reminder that, unless stated otherwise, **class fields should be private while class methods should be public**.

Requirements

The features described below must be in your program:

- Tenant class
  - It has 3 fields, all of which are public: name, which is a string, rent, which is a double, and age, which is an integer
  - **Tenant(String, int, double)**: This overloaded constructor assigns its arguments to the appropriate fields in the object.
  - **String toString()**: Returns a string in the following format, where X is the renter's name, Y is their age, and Z is their rent:

    X (Y) | $Z/mo

- Building class
  - It has 4 fields:
    1. An array of Tenants called "apartments"
       - Note that an apartment is considered to be empty if its value is currently null, and occupied otherwise
    2. A static double called "balance", initialized at 0
    3. A static integer called "nextId", initialized at 1
    4. An integer called id
  - **Building(int)**: This overloaded constructor initializes apartments with the size passed in the arguments. It then assigns nextId to id and increments nextId by 1.
  - A getter for the id field and for the balance field
    1. Note that, since balance is static, the getter for it must also be static
  - **int countTenants()**: Returns the number of tenants in this building. Empty apartments should not be counted.

- **boolean addTenant(Tenant, int):** If the provided integer is either out of bounds for the apartments field, or if a Tenant is already at that apartment, return false. Otherwise, assign the Tenant in the arguments to apartments at the index provided in the arguments and return true.
- **boolean removeTenant(int)**: If the provided integer is either out of bounds for the apartments field, or if that apartment is already empty, return false. Otherwise, set said to be empty and return true.
- **String listTenants():** Return a string listing all Tenants at that building, in the following format:

  Apartment X: Y

  Where X is the apartment's index in the array. If the apartment has a Tenant, replace Y with the Tenant's toString(), otherwise, list it as "Empty". Each apartment must be in its own line.

- **Tenant getTenant(int):** If the provided integer is out of bounds for the apartments field, return null. Otherwise, return the Tenant in apartments at the index provided in the argument.
- **static void collectRent(Tenant):** Increment the balance field with the value of the rent field of the Tenant in the argument.
- **static void payForService(double)**: Decrement the balance field with the value provided in the argument.

- Driver:
  - Create an Arraylist of Buildings called "buildings"
  - In a loop, implement the menu options below:
    1. **Add building**: Prompts the user for the number of apartments in that building. Creates said building with the information entered, adding the building to the arraylist.
    2. **Remove building**: Prompts the user for a building id. If there exists a building with said id in the arraylist, remove it and print a message saying it has been removed. Otherwise, print a message saying that no such building exists.
    3. **List all buildings:** Prints the following information, one building per line, for all buildings in the arraylist:

       Building X: Y tenants

       Where X is the building's id and Y is its current occupancy

    4. **List all tenants:** Prompts the user for a building id. If there exists a building with said id in the arraylist, print all of its tenants (using their

toString()). Otherwise, print a message saying that no such building exists.

5. **Add tenant:** Prompts the user for a building id. If no building with said id exists, print an error message. Otherwise, print all tenants in that building and then prompt the user for the tenant's information (name, age, rent). Afterwards, prompt the user for the apartment number where the tenant will be added. If said apartment already contains a Tenant, print an error message. Otherwise, add the newly created Tenant to the selected apartment.
6. **Remove tenant:** Prompts the user for a building id. If no building with said id exists, print an error message. Otherwise, print all tenants in that building and prompt the user for the apartment number which should have its tenant removed. If the apartment is empty or invalid, print an error message. Otherwise, remove the tenant from said apartment.
7. **Update Tenant:** Prompts the user for a building id. If no building with said id exists, print an error message. Otherwise, print all tenants in that building and prompt the user for the apartment number of the tenant to be updated. If the apartment is empty or invalid, print an error message. Otherwise, print the tenant's current information and prompt the user for the updated information (name, age, and rent). Update the Tenant's fields with new information provided.
8. **Charge rent:** Prompts the user for a building id. If no building with said id exists, print an error message. Otherwise, print all tenants of that building and prompt the user for the apartment number of the tenant to be charged. If the apartment is empty or invalid, print an error message. Otherwise, retrieve the Tenant at that index and use the appropriate Building method to update the system's balance.
9. **Pay for service:** Prints out the current balance in the system and then prompts the user for the amount changed for the service. Uses the appropriate Building method to update the balance, printing out the new balance.
0. **Quit**: Terminates the program

Deliverables

- Assignment2.java (driver)
- Building.java
- Tenant.java

Considerations

- This assignment may seem intimidating, but that's just because of the number of things you have to do in the driver. Don't forget that you will get partial credit for partial work.
- Feel free to add any helper methods which you believe are necessary, but know you will be not graded for them.
- Despite what the deliverables say above, you can place all of your classes in a single file
    - You will notice that placing all of your code in a single file, while still valid, will make your single file very large and potentially harder to navigate. In general, you will want to place different classes on different files.
- For variables of type double, you will **not** lose points due to rounding errors. If the sample output below says "1.00001" and yours says "1.00002", you will still get full credit **as long as your logic is correct.** Similarly, you do **not** need to round off your numbers to a specific number of decimal places. If the sample output says "2.75" and yours says "2.747155482733", you will still get full credit **as long as your logic is correct.**
- Note that when you initialize an array, all of its indexes will be initialized with their default values. This default value is 0 for numbers, false for Booleans, the null character for chars, and null for complex datatypes.
- Usually, you would not want to have the fields of the Tenant class public, as that allows outside code to enter "invalid" values to Tenant objects (e.g.: set a Tenant's rent to be a negative number). This is being done so you can practice accessing public fields in objects, and also to reduce the amount of code you have to write.

Sample Output (user input in red)

```
[Apartment Complex Management System]
1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 1

How many tenants can this building have? 5
```

```
Building 1 created

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 1

How many tenants can this building have? 7
Building 2 created

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 1

How many tenants can this building have? 3
Building 3 created

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
```

```
Select option: 3

Building 1: 0 tenants.
Building 2: 0 tenants.
Building 3: 0 tenants.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 5

Enter id of building: 1
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Empty
Apartment 4: Empty

Enter tenant name: Alice
Enter tenant age: 60
Enter tenant rent: $300.50
Add tenant to which apartment? 3
Tenant added.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 5
```

```
Enter id of building: 1
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Alice (60) | $300.50/mo
Apartment 4: Empty

Enter tenant name: Bob
Enter tenant age: 37
Enter tenant rent: $450
Add tenant to which apartment? 4
Tenant added.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 5

Enter id of building: 1
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Alice (60) | $300.50/mo
Apartment 4: Bob (37) | $450.00/mo

Enter tenant name: Charlie
Enter tenant age: 47
Enter tenant rent: $350.75
Add tenant to which apartment? 3
There is already a tenant at that apartment.

1. Add building
2. Remove building
3. List all buildings
```

```
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 5

Enter id of building: 2
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Empty
Apartment 4: Empty
Apartment 5: Empty
Apartment 6: Empty

Enter tenant name: David
Enter tenant age: 58
Enter tenant rent: $200.55
Add tenant to which apartment? 0
Tenant added.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 7

Enter id of building: 1
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Alice (60) | $300.50/mo
Apartment 4: Bob (37) | $450.00/mo
```

```
Update tenant at which apartment? 2
That is not a valid apartment.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 7

Enter id of building: 1
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Alice (60) | $300.50/mo
Apartment 4: Bob (37) | $450.00/mo

Update tenant at which apartment? 3
Current tenant information:
Alice (60) | $300.50/mo
Enter new name: Alice
Enter new age: 60
Enter new rent: $290.50
Tenant updated.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 6
```

```
Enter id of building: 1
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Alice (60) | $290.50/mo
Apartment 4: Bob (37) | $450.00/mo

Remove tenant at which apartment? 4
Tenant removed.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 8

Enter id of building: 2
Apartment 0: David (58) | $200.55/mo
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Empty
Apartment 4: Empty
Apartment 5: Empty
Apartment 6: Empty

Charge rent of which tenant? 0
Tenant charged. New balance is $200.55

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
```

```
8. Charge rent
9. Pay for service
0. Quit
Select option: 9

Current balance is $200.55
Enter service amount: $200
New balance is $0.55

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 3

Building 1: 1 tenants.
Building 2: 1 tenants.
Building 3: 0 tenants.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 2

Enter id of building to remove: 3
Building removed.

1. Add building
2. Remove building
```

```
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 1

How many tenants can this building have? 10
Building 4 created

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 3

Building 1: 1 tenants.
Building 2: 1 tenants.
Building 4: 0 tenants.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 4

Enter id of building to list: 3
```

```
No building with such id.

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 4

Enter id of building to list: 2
Apartment 0: David (58) | $200.55/mo
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Empty
Apartment 4: Empty
Apartment 5: Empty
Apartment 6: Empty

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 4

Enter id of building to list: 1
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Alice (60) | $290.50/mo
Apartment 4: Empty
```

```
1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 4

Enter id of building to list: 4
Apartment 0: Empty
Apartment 1: Empty
Apartment 2: Empty
Apartment 3: Empty
Apartment 4: Empty
Apartment 5: Empty
Apartment 6: Empty
Apartment 7: Empty
Apartment 8: Empty
Apartment 9: Empty

1. Add building
2. Remove building
3. List all buildings
4. List all tenants
5. Add tenant
6. Remove tenant
7. Update tenant
8. Charge rent
9. Pay for service
0. Quit
Select option: 0

Shutting off...
```