Answers Tut 2

Indian Institute of Information Technology, Allahabad

B.Tech. III Sem. (Object Oriented Methodology)

Tutorial-Assignments-2

SEE Below For Answers

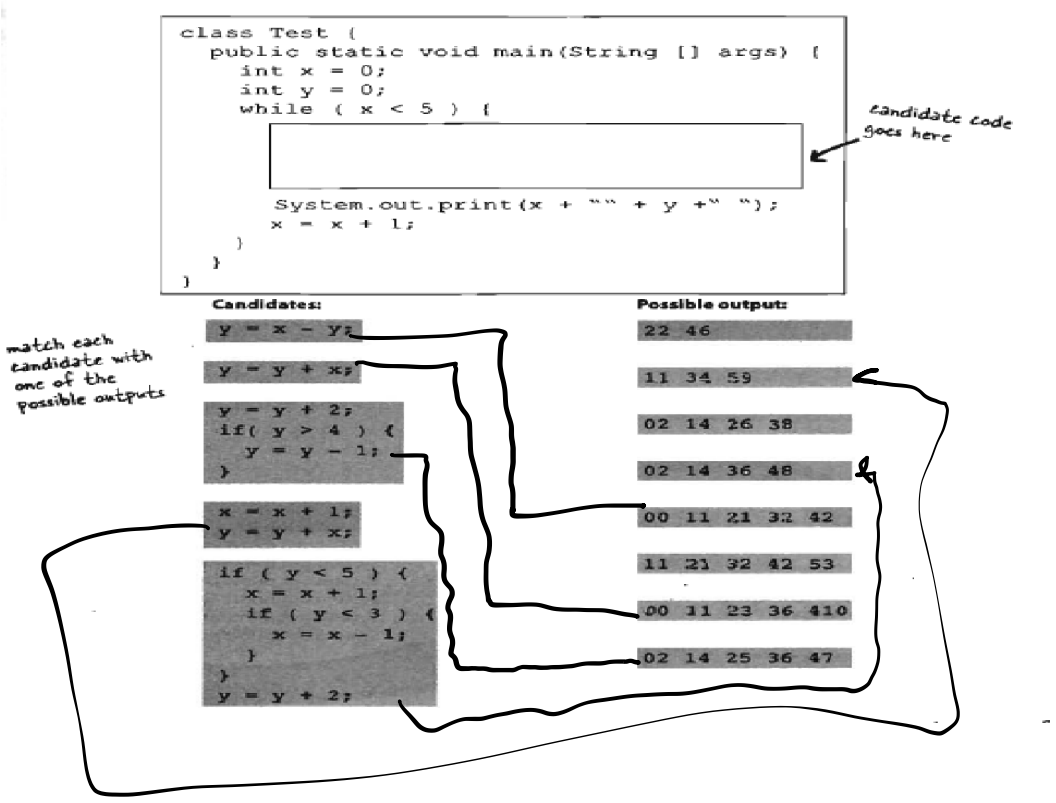
Date of Tutorial: 16/08 /2023 **Instructors: Prof. O. P. Vyas , Dr. Ranjana Vyas**

Ques 01:

Determine the various classes and methods you would require to design an application that can be used by the college for elective subject allotment. Professors are allotted the subjects that they will be teaching. The courses will be either Core Course or an Elective one. The students will be given a list of possible elective subjects based on their course. A student cannot choose a subject that they have been allotted in a previous semester. Students can provide their preference order of electives and they are then allotted an elective based on their CGPA. Additional features can be added to this application to provide information such as the number of students allotted to a subject. Classify the methods determined as **static or instance**.

Identify various OO Characteristics like Inheritance that you may observe in Class-Object identification in the given problem domain. What DATA and METHODS will be inherited from which Parent Class and what advantages it would bring?

Ques 02: A short Java program is listed below with one block of program missing. Your task is to match the candidate block of the code with suitable output that would be seen if the block were inserted. All the output shown will be used, some more than once.



Que. 03: When designing a class to describe an object, how are the behaviors of the object

Program 01:

```
public class Team
```

represented in the class? Use the following BaseballGame and Team classes to answer the ensuing questions.

a) After the following statement, what is the value of each field of the game object?
BaseballGame game = new BaseballGame();
b) After the following statement, what is the value of each field of the giants object?
Team giants = new Team();

Consider the following statements when answering next questions (d) and (e):

Team angels;
BaseballGame worldSeries;

c) How many objects are there in memory after the previous two statements are done executing?
d) Which consumes more memory: angels or worldSeries? Justify your answer?

```
{   public String name;
public String city;
public int numberOfWins, numberOfLosses;  }
public class BaseballGame
{
public Team home, visitor;
public int homeScore, visitorScore;
public void homeTeamScored(int numberOfRuns)
{
homeScore += numberOfRuns;
}
public void visitorTeamScored(int numberOfRuns)
{
visitorScore += numberOfRuns;      }
public void gameOver()
{
if(homeScore > visitorScore)
{
home.numberOfWins++;
visitor.numberOfLosses++;
}
else
{
visitor.numberOfWins++;
home.numberOfLosses++;
}    }
public void setHomeTeam(Team t)
{   home = t;    }
public void setVisitingTeam(Team visitor)
{    visitor = visitor;    }  }
```

Ques 04 : Problem Statement for designing object-oriented software-simulator application

A company intends to build a two-floor office building and equip it with an elevator. The company wants you to develop an object-oriented software-simulator application in Java that models the operation of the elevator to determine whether it will meet the company's needs. The company wants the simulation to contain an elevator system. The application consists of three parts. The first and most substantial part is the simulator, which models the operation of the elevator system. The second part is the display of this model on screen so that the user may view it graphically. The final part is the graphical user interface, or GUI, that allows the user to control the simulation.

The elevator system consists of an elevator shaft and an elevator car. In our simulation, we model people who ride the elevator car (referred to as "the elevator") to travel between the floors in the elevator shaft, as shown in Fig **1, Fig 2, Fig 3.**

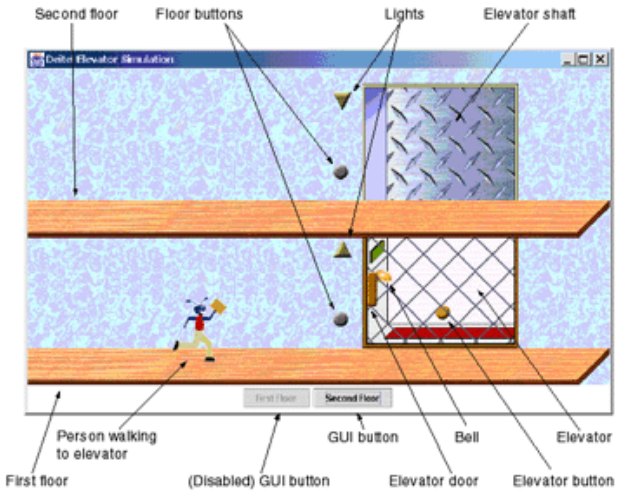


Figure 1 : Person moving towards elevator on the first floor.

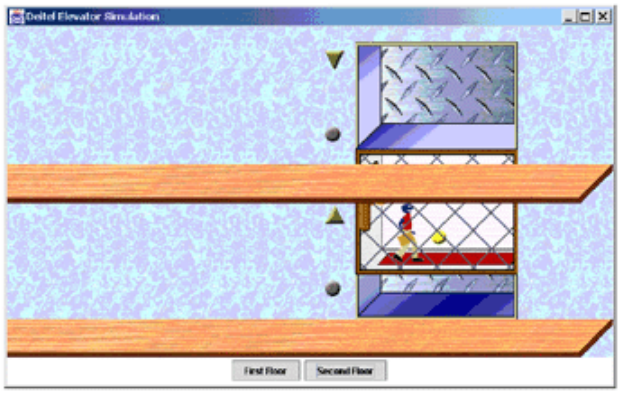


Figure 2 : Person riding the elevator to the second floor.

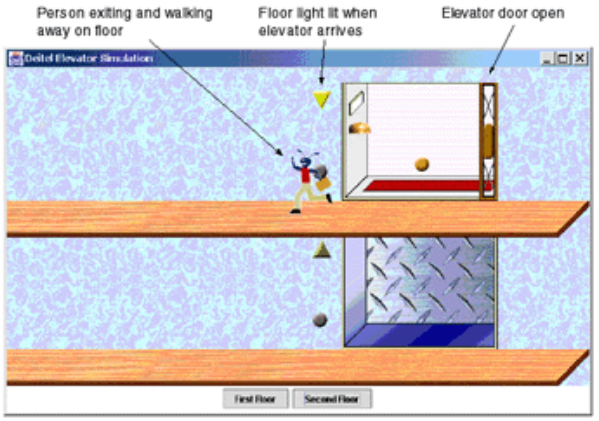


Figure 3 : Person walking away from elevator.

Preliminary Conditions: 1- The elevator contains a door (called the "elevator door") that opens upon the elevator's arrival at a floor and closes upon the elevator's departure from that floor. The elevator door is closed during the trips between floors to prevent the passenger from being injured by brushing against the wall of the elevator shaft. In addition, the elevator shaft connects to a door on each floor (referred to as the two "floor doors"), so people cannot fall down the shaft when the elevator is not at a floor. Note that we do not display the floor doors in the figures, because they would obscure the inside of the elevator (we use a mesh door to represent the elevator door because mesh allows us to see inside the elevator). The floor door opens

concurrently with the elevator door, so it appears as if both doors open at the same time. A person sees only one door, depending on that person's location. When the person is inside the elevator, the person sees the elevator door and can exit the elevator when this door opens; when the person is outside the elevator, the person sees the floor door and can enter the elevator when that door opens.

Preliminary Conditions: 2- The elevator starts on the first floor with all the doors closed. To conserve energy, the elevator moves only when necessary. For simplicity, the elevator and floors each have a capacity of only one person.

Preliminary Conditions: 3- The user of our application should, at any time, be able to create a unique person in the simulation and situate that person on either the first or second floor (Fig 1). When created, the person walks across the floor to the elevator. The person then presses a button on the floor next to the elevator shaft (referred to as a "floor button"). When pressed, that floor button illuminates, then requests the elevator. When summoned, the elevator travels to the person's floor. If the elevator is already on that person' floor, the elevator does not travel. Upon arrival, the elevator resets the button inside the elevator (called the "elevator button"), sounds the bell inside the elevator, then opens the elevator door (which opens the floor door on that floor). The elevator then signals the elevator shaft of the arrival. The elevator shaft, upon receiving this message, resets the floor button and illuminates the light on that floor.

Preliminary Conditions: 4- - Occasionally, a person requests the elevator when it is moving. If the request was generated at the floor from which the elevator just departed, the elevator must "remember" to revisit that floor after carrying the current passenger to the other floor.

Preliminary Conditions: 5- - When the floor door opens, the person enters the elevator after the elevator passenger (if there is one) exits. If a person neither enters nor requests the elevator, the elevator closes its door and remains on that floor until the next person presses a floor button to summon the elevator. When a person enters the elevator, that person presses the elevator button, which also illuminates when pressed. The elevator closes its door (which also closes the floor door on that floor) and moves to the opposite floor. The elevator takes five seconds to travel between floors. When the elevator arrives at the destination floor, the elevator door opens (along with the floor door on that floor) and the person exits the elevator.

Preliminary Conditions: 6- - The application user introduces a person onto the first or second floor by pressing the First Floor button or the Second Floor button, respectively. When the user presses the First Floor button, a person should be created (by the elevator simulation) and positioned on the first floor of the building. When the user presses the Second Floor button, a person should be created and positioned on the second floor. Over time, the user can create any number of people in the simulation, but the user cannot create a new person on an occupied floor. For example, Fig 1 shows that the First Floor button is disabled to prevent the user from creating more than one person on the first floor. Figure 2 shows that this button is reenabled when the person rides the elevator.

Preliminary Conditions: 7- - The company requests that audio be integrated into the simulation. For example, as a person walks, the application user should hear the footsteps. Each time a floor or elevator button is pressed or reset, the user should hear a click. The bell should ring upon the elevator's arrival, and doors should creak when they open or close. Lastly, "elevator music" should play as the elevator travels between floors.

Q1. UML diagram attached

→ All methods used are instance methods

→ Advantages

→ UML diagram depicts everything.

→ This allows for an OO design

→ This allows for flexibility, reusability and easy maintenance and updation of code

→ Clearly defined classes and subclasses.

→ Inheritance helps in differentiating 2 classes which have many common characteristics.

Q2 Done above

Q3. Behaviors of objects are represented through methods defined in class.

a) game → home → city = NULL
→ numOfUnity = 0
num of losses = 0

visitor → city = NULL
→ numOfUnity = 0
num of losses = 0

homeScore = 0
visitorScore = 0

b) giants → city = NULL
→ numOfUnity = 0
num of losses = 0

c) 0 as they reference variables are declared..

d) Both consume same memory as they are only referencing [32 bit integers].

Q4. UML Diagram attached.

Self explanatory