OBJECT ORIENTED PROGRAMMING

OOP OVERVIEW

INTRODUCTION

- Object-Oriented Programming (OOP) is the term used to describe a programming approach based on objects and classes.
- Allows us to organize software as a collection of objects that consist of both data and behavior.

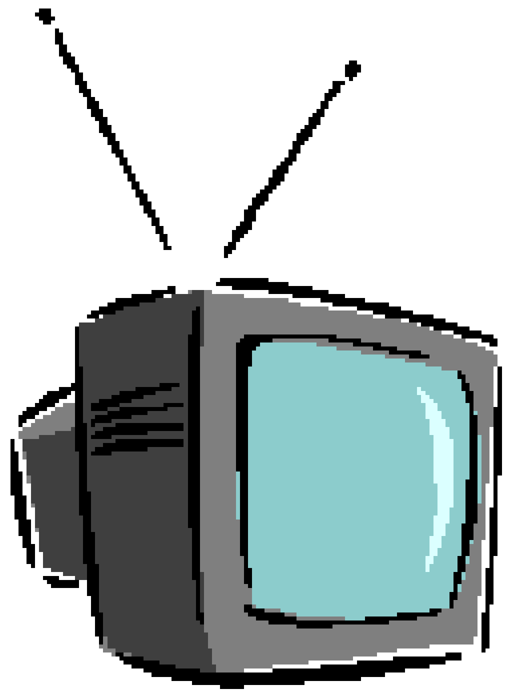
OOP ENCOURAGE

- **Modularization:** where the application can be decomposed into modules.
- **Software re-use:** where an application can be composed from existing and new modules.

OOP FEATURES

- **Classes**
- **Objects**
- **Classification**
- **Encapsulation**
- **Abstraction**
- **Polymorphism**
- **Inheritance**

FEATURES AND PROPERTIES OF A TV



FEATURES AND PROPERTIES

- We do not have to open the case to use it.
- We have some controls to use it (buttons on the box, or a remote control).
- We can still understand the concept of a television, even if it is connected to a DVD player.
- It is complete when we purchase it, with any external requirements.
- Well documented.
- The TV will not crash!!

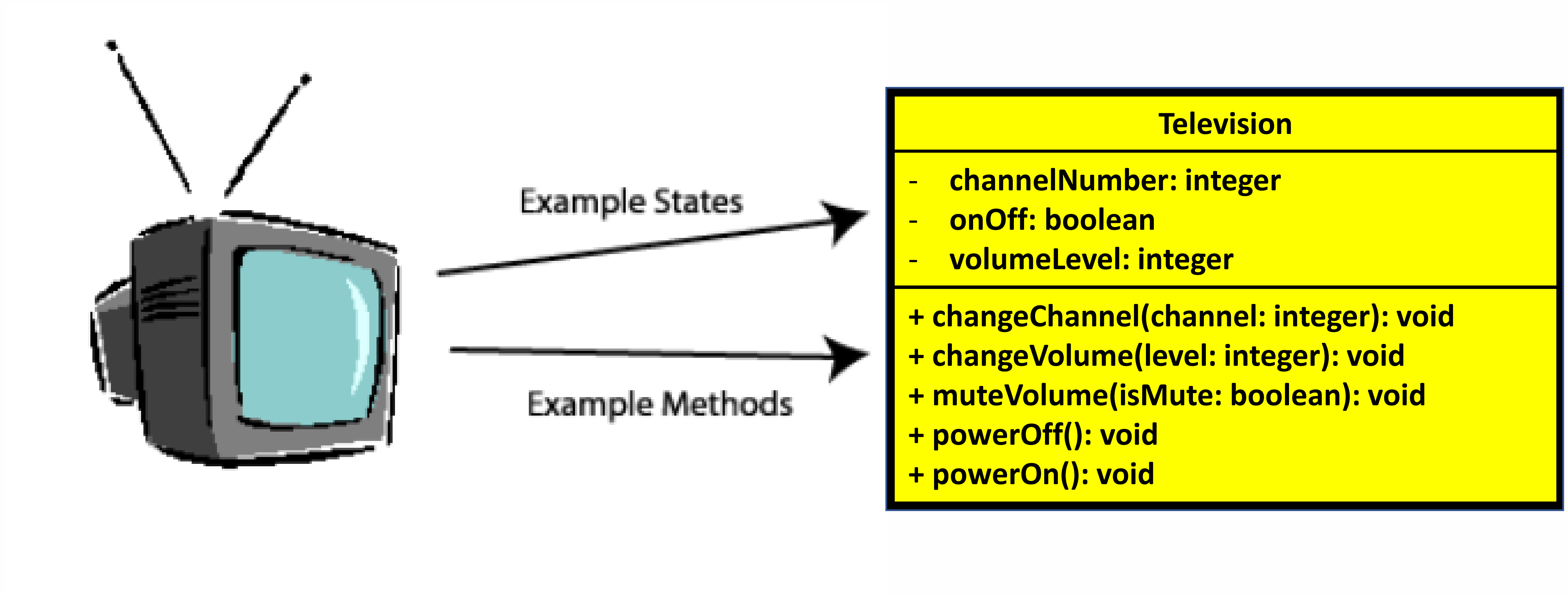
THE CONCEPT OF A CLASS (AS A TV)

- Provide a well-defined interface - such as the remote control of the television.
- Represent a clear concept - such as the concept of a television.
- Be complete and well-documented - the television should have a plug and should have a manual that documents all features.
- The code should be robust - it should not crash, like the television.

THE CONCEPT OF A CLASS

- Classes allow us a way to represent complex structures within a programming language.
- They have two components:
  - **States (or data):** are the values that the object has.
  - **Methods (or behaviour):** are the ways in which the object can interact with its data, the actions.

A TV AS A CLASS



Television

- channelNumber: integer
- onOff: boolean
- volumeLevel: integer

+ changeChannel(channel: integer): void
+ changeVolume(level: integer): void
+ muteVolume(isMute: boolean): void
+ powerOff(): void
+ powerOn(): void

ABSTRACT DATA TYPE

- Collection of *objects* (or ***values***) and a corresponding set of ***methods***.
- Is implemented via a Class.

CLASS

- A class is an **extensible program-code-template** for creating **objects**, providing initial values for **state** (member variables) and implementations of **behavior** (member functions or methods)[1].
- Is the **blueprint** from which individual **objects** are created[3].
- Is an Abstract Data Type.

OBJECT

- An object stores its state in **fields** (variables in some programming languages) and exposes its behavior through **methods** (functions in some programming languages) [4].
- A **variable**, a **data structure**, or a **function**, and as such, is a **location in memory** having a **value** and possibly **referenced** by an **identifier**[2].
- A particular **instance** of a class[2].

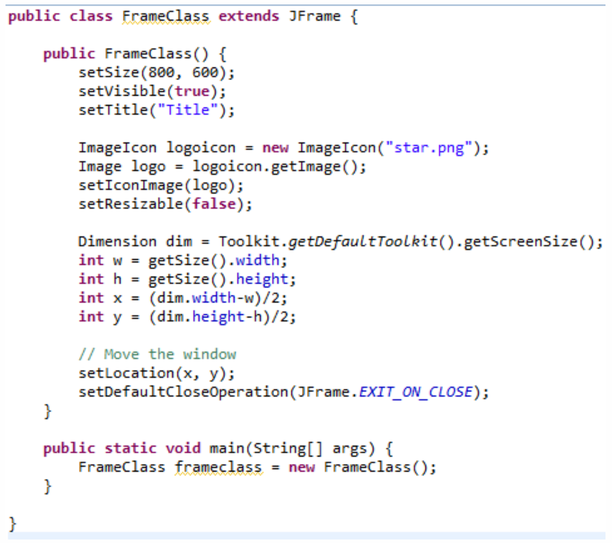
OBJECT

- Encapsulation of data.
- Real-world objects share two characteristics:
  - They all have **state** and **behavior**[4].
- On object also has a **identity** (a unique **reference**)

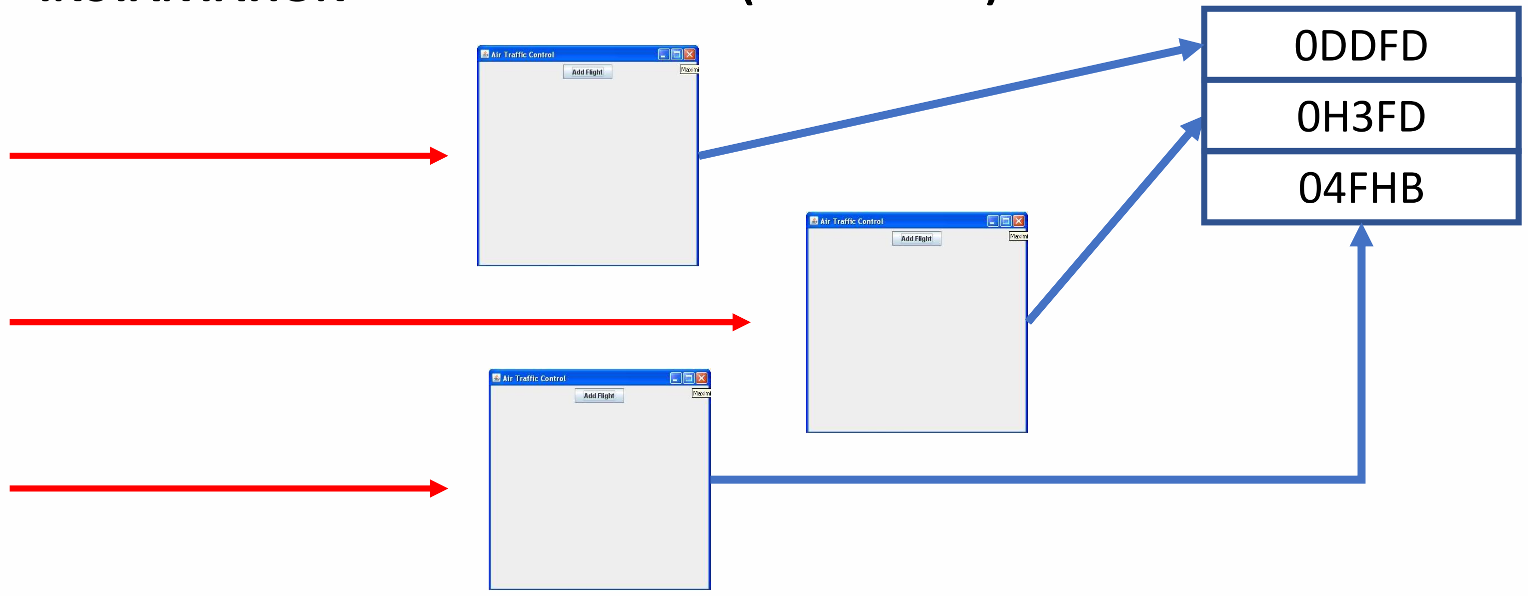
INSTANCE

- Is a **concrete occurrence** of any **object**, existing usually during the runtime of a computer program[1].

CLASS **INSTANTATION** **OBJECTS (INSTANCES)** **REFERENCES (MEMORY)**

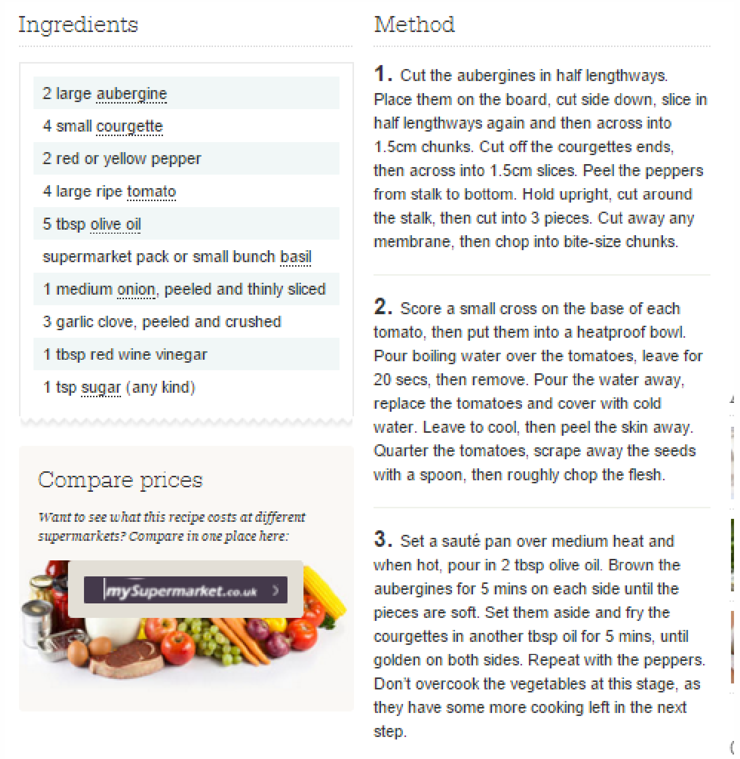


```
public class FrameClass extends JFrame {

    public FrameClass() {
        setSize(800, 600);
        setVisible(true);
        setTitle("Title");

        ImageIcon logoicon = new ImageIcon("star.png");
        Image logo = logoicon.getImage();
        setIconImage(logo);
        setResizable(false);

        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        int w = getSize().width;
        int h = getSize().height;
        int x = (dim.width-w)/2;
        int y = (dim.height-h)/2;

        // Move the window
        setLocation(x, y);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        FrameClass frameclass = new FrameClass();
    }

}
```



CLASS - OBJECT

CLASS



Ingredients

- 2 large aubergine
- 4 small courgette
- 2 red or yellow pepper
- 4 large ripe tomato
- 5 tbsp olive oil
- supermarket pack or small bunch basil
- 1 medium onion, peeled and thinly sliced
- 3 garlic clove, peeled and crushed
- 1 tbsp red wine vinegar
- 1 tsp sugar (any kind)

Compare prices

Want to see what this recipe costs at different supermarkets? Compare in one place here:

mySupermarket.co.uk

Method

1. Cut the aubergines in half lengthways. Place them on the board, cut side down, slice in half lengthways again and then across into 1.5cm chunks. Cut off the courgettes ends, then across into 1.5cm slices. Peel the peppers from stalk to bottom. Hold upright, cut around the stalk, then cut into 3 pieces. Cut away any membrane, then chop into bite-size chunks.

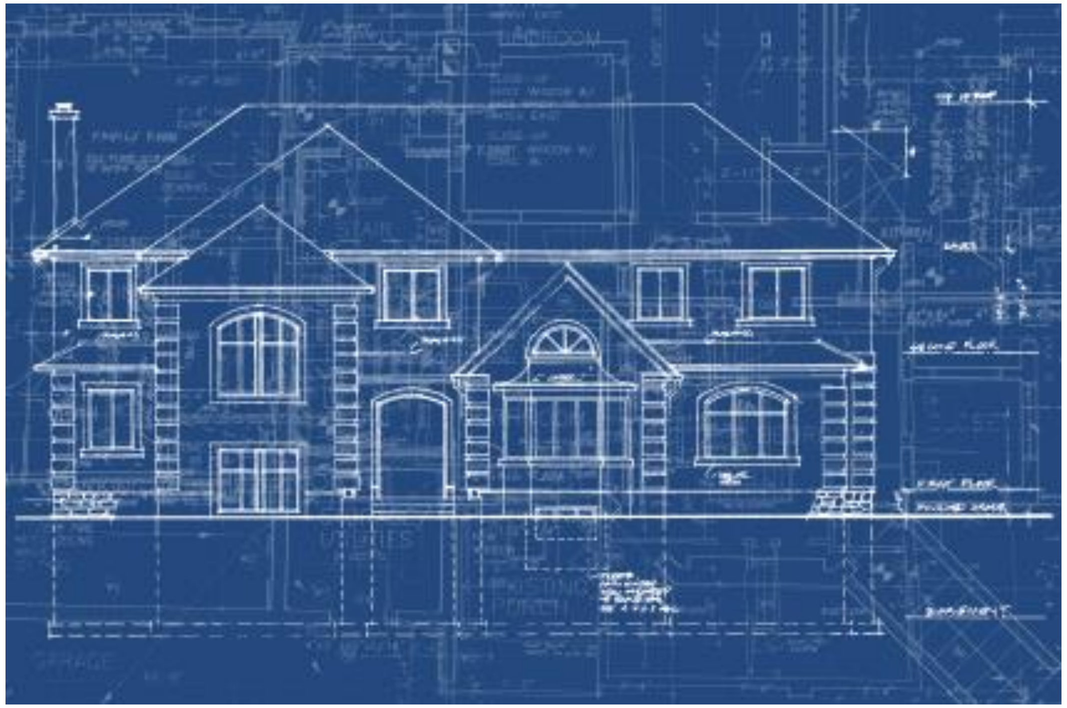
2. Score a small cross on the base of each tomato, then put them into a heatproof bowl. Pour boiling water over the tomatoes, leave for 20 secs, then remove. Pour the water away, replace the tomatoes and cover with cold water. Leave to cool, then peel the skin away. Quarter the tomatoes, scrape away the seeds with a spoon, then roughly chop the flesh.

3. Set a sauté pan over medium heat and when hot, pour in 2 tbsp olive oil. Brown the aubergines for 5 mins on each side until the pieces are soft. Set them aside and fry the courgettes in another tbsp oil for 5 mins, until golden on both sides. Repeat with the peppers. Don't overcook the vegetables at this stage, as they have some more cooking left in the next step.

OBJECT

CLASS - OBJECT

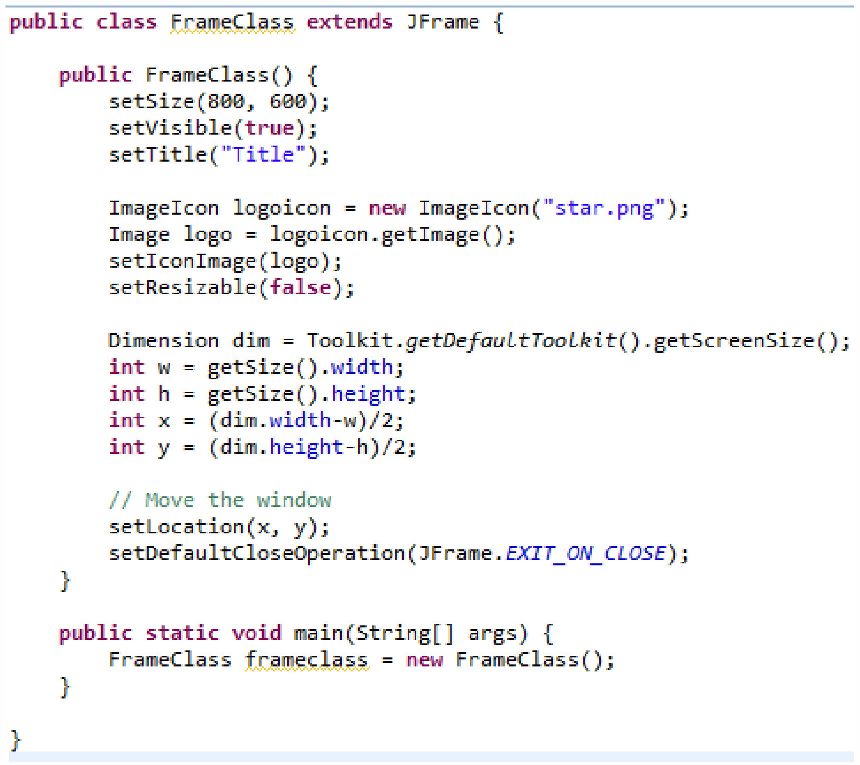
CLASS



OBJECT

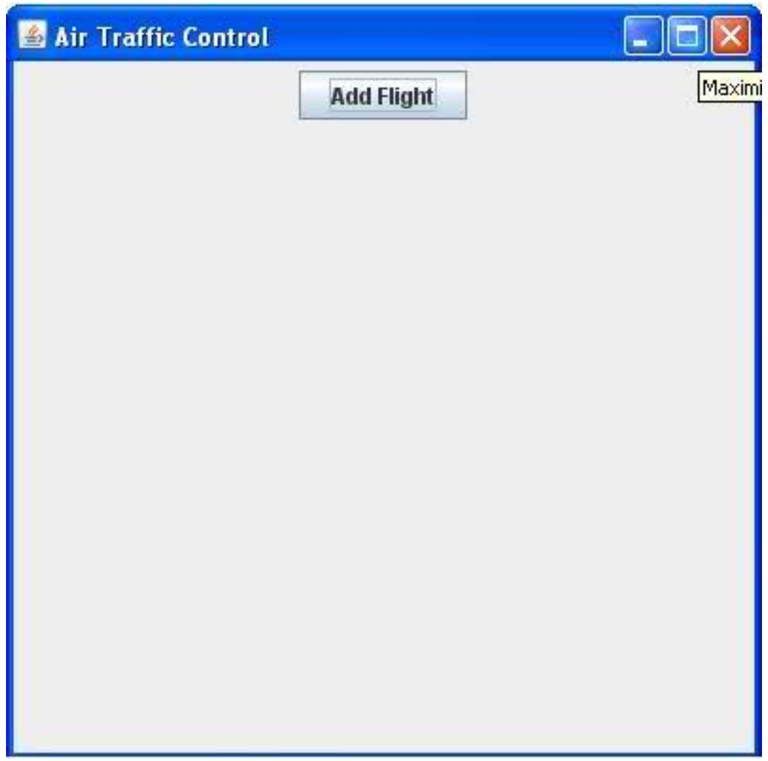
CLASS - OBJECT

CLASS



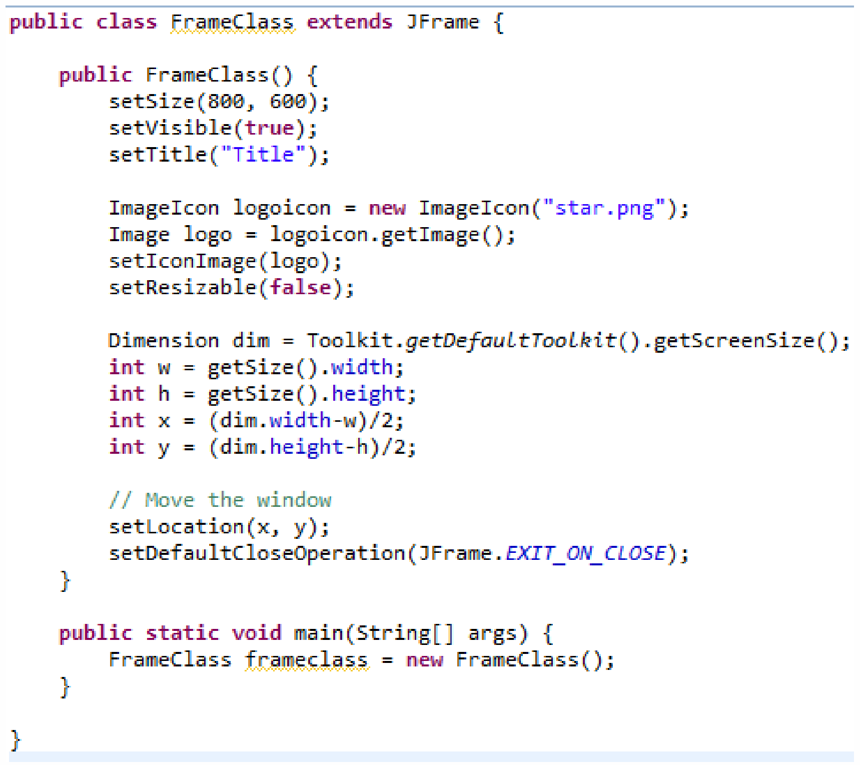
```
public class FrameClass extends JFrame {

    public FrameClass() {
        setSize(800, 600);
        setVisible(true);
        setTitle("Title");

        ImageIcon logoicon = new ImageIcon("star.png");
        Image logo = logoicon.getImage();
        setIconImage(logo);
        setResizable(false);

        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        int w = getSize().width;
        int h = getSize().height;
        int x = (dim.width-w)/2;
        int y = (dim.height-h)/2;

        // Move the window
        setLocation(x, y);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        FrameClass frameclass = new FrameClass();
    }

}
```

OBJECT



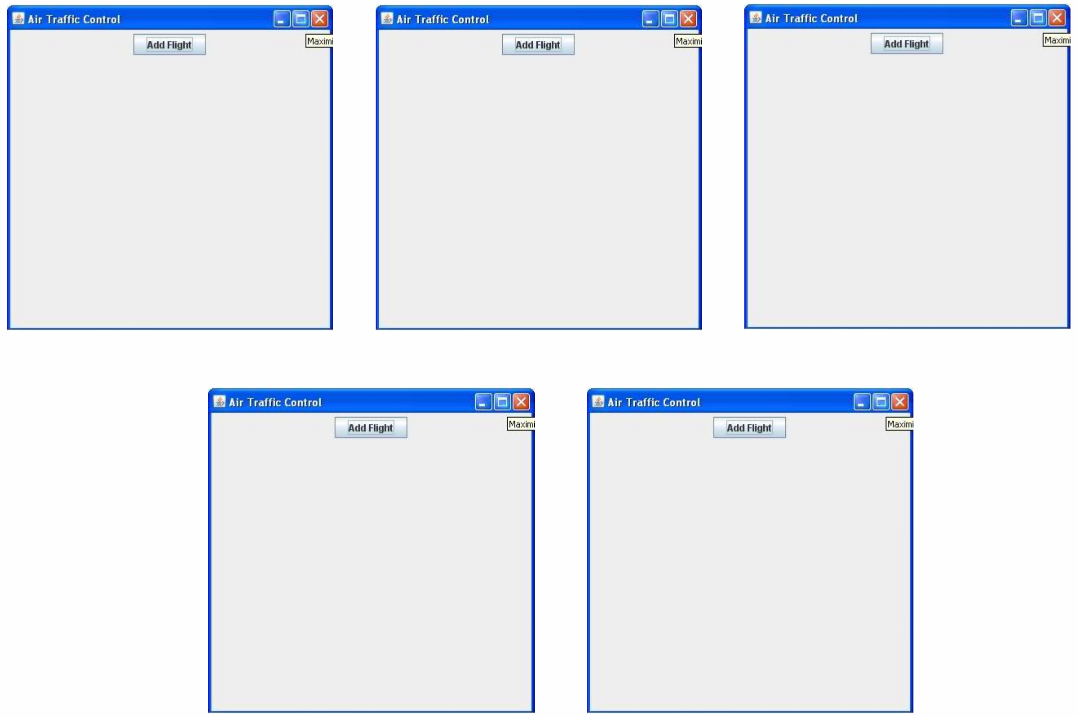
CLASS - OBJECT

CLASS



```
public class FrameClass extends JFrame {

    public FrameClass() {
        setSize(800, 600);
        setVisible(true);
        setTitle("Title");

        ImageIcon logoicon = new ImageIcon("star.png");
        Image logo = logoicon.getImage();
        setIconImage(logo);
        setResizable(false);

        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        int w = getSize().width;
        int h = getSize().height;
        int x = (dim.width-w)/2;
        int y = (dim.height-h)/2;

        // Move the window
        setLocation(x, y);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        FrameClass frameclass = new FrameClass();
    }

}
```

OBJECTS


Air Traffic Control
Add Flight
Maximi