UNTAR
Universitas Tarumanagara

Terakreditasi BAN PT A (Unggul)

UNTAR untuk INDONESIA

Object-based Programming

Week 2 – Object Oriented Programming

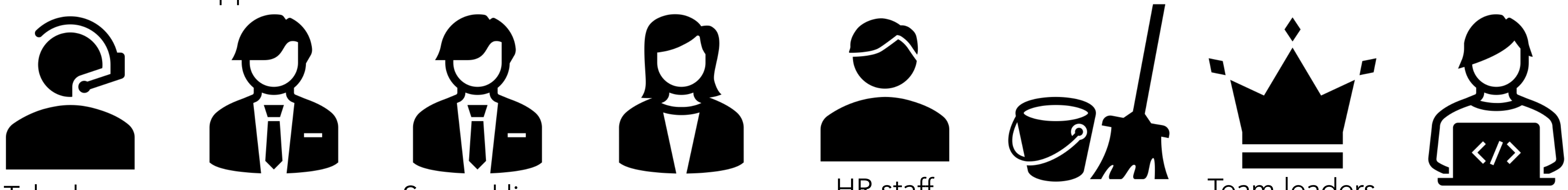
Procedural Programming

```
// main.cpp

#include “rectangle.h”
#include “circle.h”

int main() {
   ...
}
```

```
// rectangle.cpp

int calculatePerimeter() { }
int calculateArea() { }
```

```
// circle.cpp

const double PI = 3.14;

double calculateArea() { }
```

Case: IT Telephone Support



Telephone operator

First line supports staff

Second line supports staff

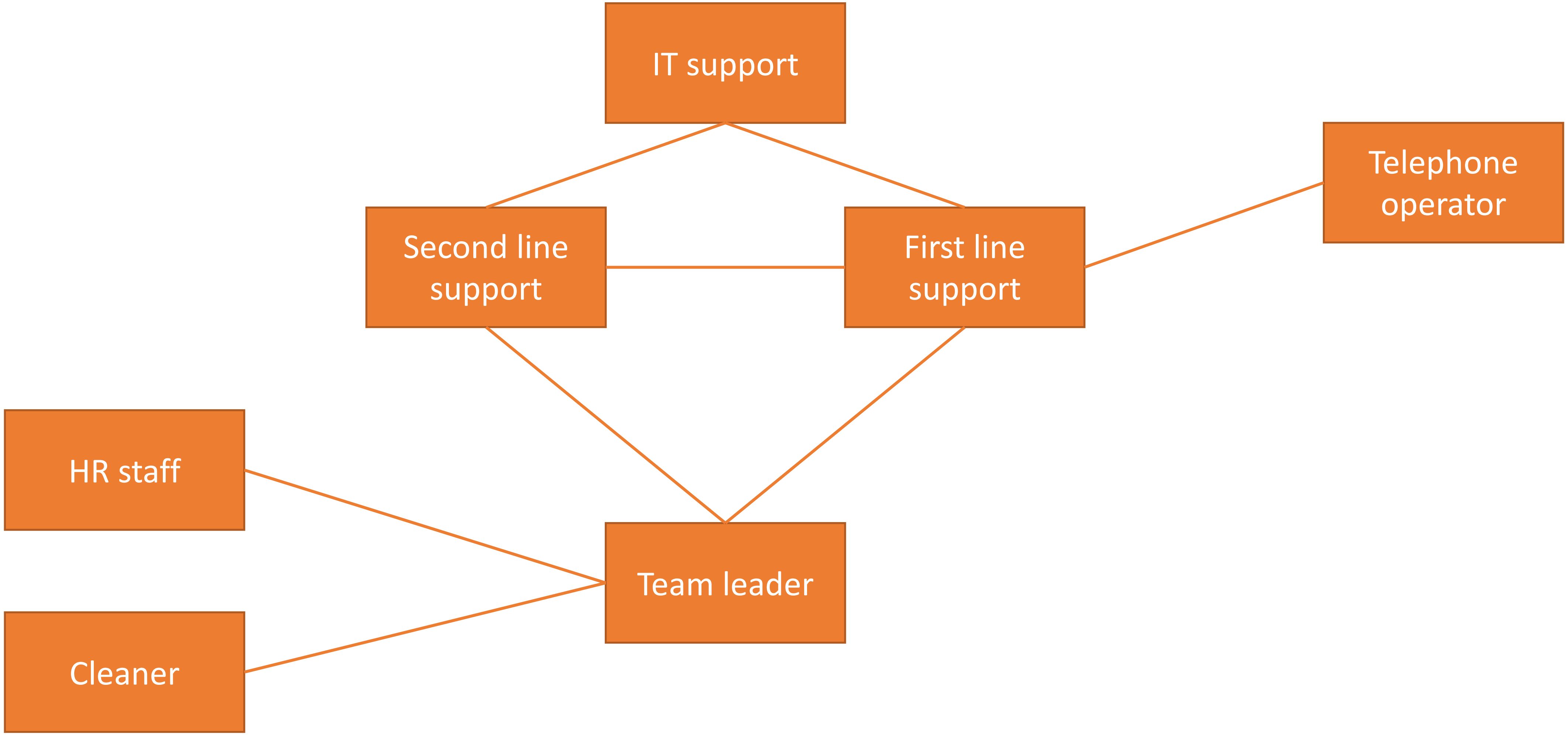
HR staff

HR staff

Cleaners

Team leaders

IT support staff



IT support

Second line support

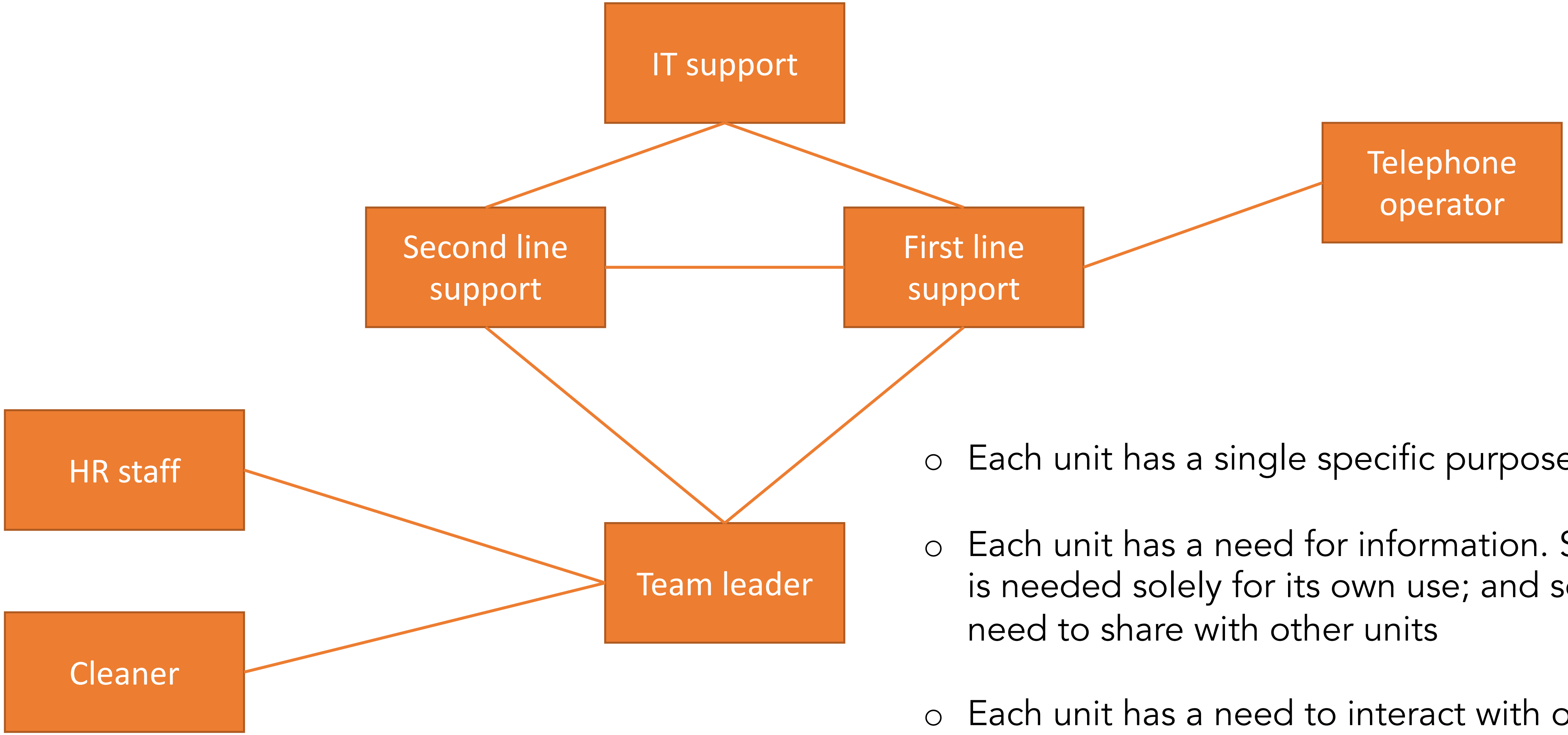
First line support

Telephone operator

HR staff

Team leader

Cleaner



IT support

Second line support

First line support

Telephone operator

HR staff

Team leader

Cleaner

- Each unit has a single specific purpose
- Each unit has a need for information. Some of it is needed solely for its own use; and some may need to share with other units
- Each unit has a need to interact with one or more other units in order to achieve some collective task

A **class** is an organisational unit of an object oriented design and program

Highly cohesive

Minimally coupled

Encapsulation

Highly cohesive

represents a single useful entity or organisational unit and does that job well

Minimally coupled

Encapsulation

Highly cohesive

Minimally coupled

limits its interactions with other classes to only those that are really necesaary for it to do what it is designed to do

Encapsulation

Highly cohesive

Minimally coupled

Encapsulation

keeps information necessary to its internal operation private and does not expose it to other classes
only makes public the information necessary for it to interact with other classes in the intended manner

Question #1

https://bit.ly/3B2zJrL

Sebuah aplikasi pemesanan tiket menggunakan OOP. Dari pilihan di bawah ini, manakah yang *class* yang mungkin digunakan?

A. Airlines

B. City

C. User

D. Flight

E. UserOrder

The anatomy of a class

The things that they “are” (the **state**)

The things that they “do” (the **behaviour**)

A sports car can be one of a variety of colours, with an engine power between 100 HP and 200 HP.

It can be a convertible or a regular model.

The car has a button that starts the engine and a parking brake.

When the parking brake is released and you press the accelerator, it drives in the direction determined by the transmission setting.

A sports car can be one of a variety of colours, with an engine power between 100 HP and 200 HP. It can be a convertible or a regular model. The car has a button that starts the engine and a parking brake. When the parking brake is released and you press the accelerator, it drives in the direction determined by the transmission setting.

State	Behaviour
Colour (text)	Press the start button
Engine power (number of HP)	Press the accelerator
Convertible? (yes/no)	
Parking brake (on/off)	

```java
public class Car {
   // Variables to describe the state
   String colour;
   double enginePower;
   boolean isConvertible;
   boolean isDirectionForwards;
   boolean parkingBrakeOn;
   boolean isMoving;
   boolean engineStarted;

   ...
}
```

```
public class Car {
   ...
   // Some methods to define the behaviour
   public void pressStartButton() {
      if (!engineStarted) {
         System.out.println(“Engine for the ” + colour +
                            ” car is starting...”);
         engineStarted = true;
      }
   }
   ...
}
```

```
public class Car {
    ...
    public void pressAccelerator() {
        if (!parkingBrakeOn && engineStarted) {
            System.out.println(“The ” + colour + “ car is moving”);
            isMoving = true;
        }
        else if (engineStarted) {
            System.out.println(“The parking brake is on!”);
            isMoving = false;
        }
        else {
            System.out.println(“The engine is not started!”);
    }
    ...
}
```

Question #2

https://bit.ly/3B2zJrL

Apa output dari pemanggilan method openBox?

A. 0

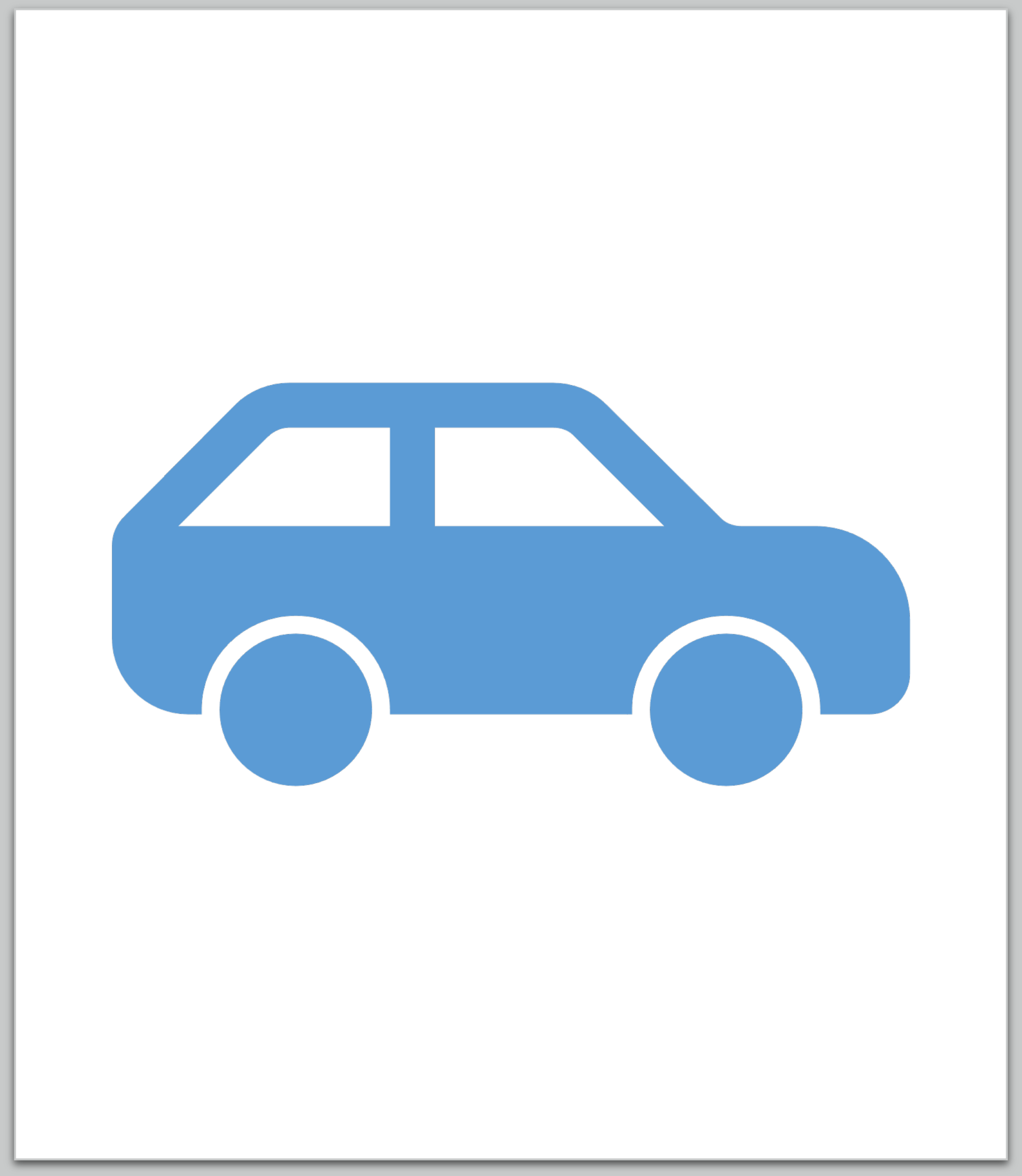
B. 1

C. 2

D. ERROR

E. content + 1

```
public class PandoraBox {
    int content = 0;

    public void openBox() {
        System.out.println(content + 1);
    }
}
```



What is the colour of the car?

```
public class Car {
    ...
    // A simple constructor method
    public Car(String colour, double enginePower, boolean isConvertible) {
        this.colour = colour;
        this.enginePower = enginePower;
        this.isConvertible = isConvertible;

        isDirectionForwards = true;
        parkingBrakeOn = true;
        isMoving = false;
        engineStarted = false;
    }
    ...
}
```

```
public class Car {
   String colour;
   double enginePower;
   boolean isConvertible;
   ...
   public Car(String colour, double enginePower, boolean isConvertible) {
      this.colour = colour;
      this.enginePower = enginePower;
      this.isConvertible = isConvertible;
      ...
   }
   ...
}

public class MainProgram {
   public static void main(String[] args) {
      Car innova = new Car(“white”, 140, false);
      innova.pressStartButton();
      innova.pressAccelerator();
      System.out.println(“The color of the car is “ + innova.colour);
   }
}
```

Car is a class
innova is an object

Class & Objects

- An object must be **declared** and **initialized**
  - Declaration
    - `Car innova`
  - Initialization (*instantiating*)
    - `innova = new Car(“white”, 140, false)`

Class & Objects

- An object must be **declared** and **initialized**
- After initialised, we can accessed the state/behaviour of the class

```
innova.pressStartButton();
```

Call the `pressStartButton()` method for the object innova

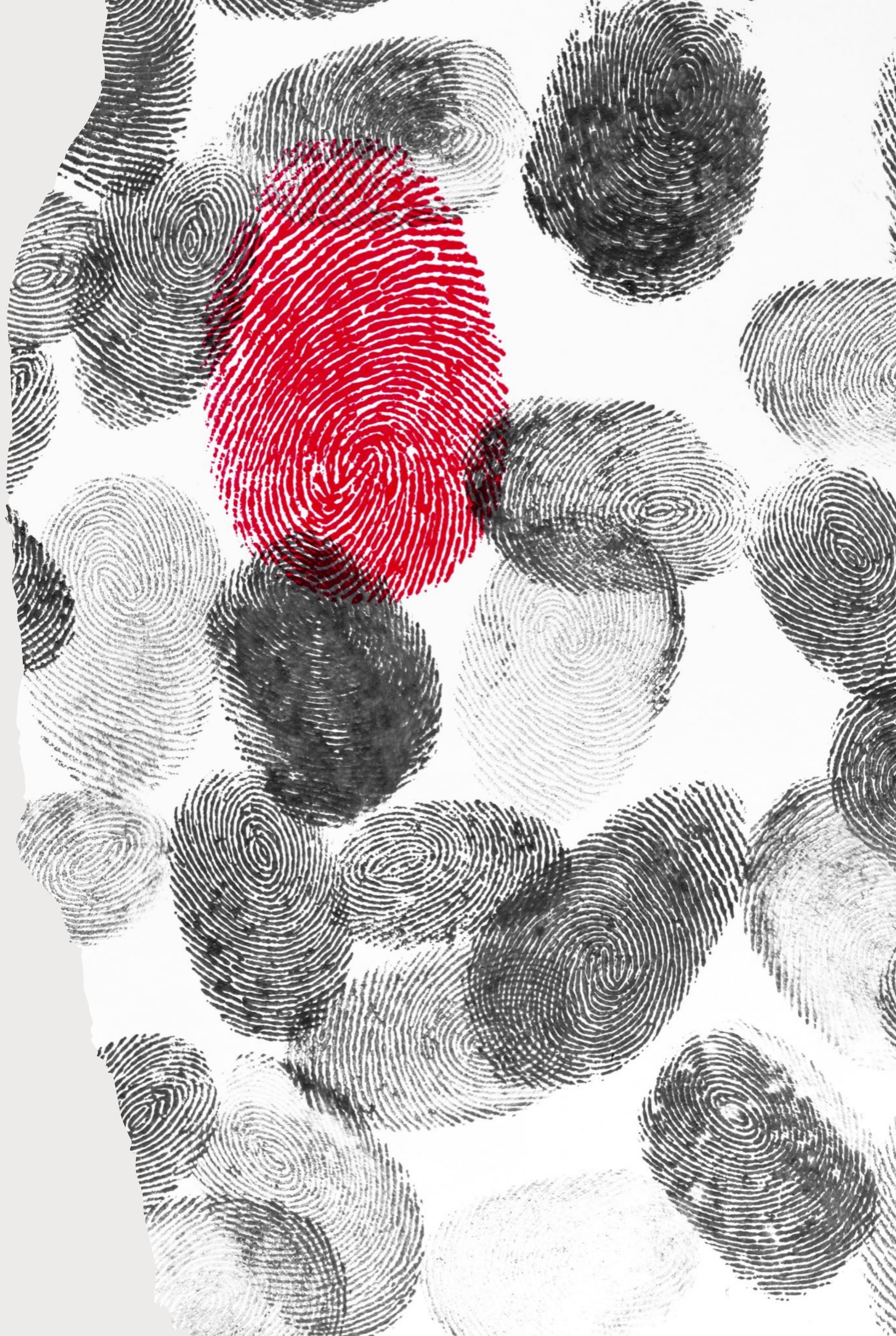
```
public class TestTrack {
   String name;
   Car c1;
   Car c2;
   Car c3;
   boolean isOpen;

   public TestTrack(String name) {
      this.name = name;
      isOpen = true;
      c1 = new Car(“red”, 100, true);
      c2 = new Car(“blue”, 120, true);
      c3 = new Car(“green”, 150, false);
   }
   ...
}
```

```
public class TestTrack {
    ...
    public void runAllCars() {
        c1.pressStartButton();
        c2.pressStartButton();
        c3.pressStartButton();

        c1.parkingBrakeOn = false;
        c2.parkingBrakeOn = false;

        c1.pressAccelerator();
        c2.pressAccelerator();
        c3.pressAccelerator();
    }
    ...
}
```

```
public class TestTrack {
   ...
   public void closeTrack() {
      c1.parkingBrakeOn = true;
      c2.parkingBrakeOn = true;
      c3.parkingBrakeOn = true;

      isOpen = false;

      System.out.println(“The test track is now closed.”);
   }
}
```

Rule of Thumb!

All data in a class should be `private` unless …



```java
public class Book {
    private String title;
    private String author;
    private int numOfPages;

    public Book() {
        this.title = “”;
        this.author = “”;
        this.numOfPages = 0;
    }

    public Book(String title, String author, int numOfPages) {
        this.title = title;
        this.author = author;
        this.numOfPages = numOfPages;
    }
}
```

```
public class Book {
   private String title;
   private String author;
   private int numOfPages;

   public Book() { ... }
   public Book(String title, String author, int numOfPages) { ... }
}

public class Main {
   public static void main(String[] args) {
      Book b = new Book(“Harry Potter 1”, “JK Rowling”, 355);
      System.out.println(b.title);
   }
}
```

ERROR

```
public class Book {
   private String title;
   private String author;
   private int numOfPages;

   public Book() { ... }
   public Book(String title, String author, int numOfPages) { ... }

   // getter
   public String getTitle() {
      return title;
   }
   public String getAuthor() { ... }
   public int getNumOfPages() { ... }

   // setter
   public void setTitle(String title) {
      this.title = title;
   }
   public void setAuthor(String author) { ... }
   public void setNumOfPages(int numOfPages) { ... }
}
```

The Reason

```
public class Student {
   public double ipk;
}

public class Main {
   public static void main(String[] args) {
      Student janson = new Student();
      janson.ipk = 1000000000;
   }
}
```

```
public class Student {
    private double ipk;

    public void setIpk(double ipk) {
        if (ipk < 0.0 || ipk > 4.0) {
            System.out.println(“ERROR: Invalid IPK value.”);
        } else {
            this.ipk = ipk;
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Student janson = new Student();
        janson.setIpk(1000000000);    // output error message and IPK unchanged
        janson.setIpk(3.6);           // success
    }
}
```