BTEC Level 3 Computing

Unit 1 - Principles of Computer Science

Object-Orientated Programming

Object-Orientated Programming

What is Object-Oriented Programming?

A programming paradigm based on "objects" containing data and code.

Organizes software design around data/objects rather than functions and logic.

Models real-world entities in code.

Promotes cleaner, more maintainable, and reusable code.

Structure Classes

Blueprint or template for creating objects.

Define attributes (data) and methods (behavior).

There are a lot of examples out there of creating vehicles or animals.

Structure Classes Example

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def start_engine(self):
        return f"The {self.brand} {self.model}'s engine is running"
```

Structure
Objects/Instances

Instances created from classes.

Represent specific examples of a class.

Contain unique data but share behavior.

Structure Objects/Instances Example

- *# Creating car objects*

```
my_car = Car("Toyota", "Corolla")
sports_car = Car("Ferrari", "F40")
print(my_car.start_engine())
print(sports_car.start_engine())
```

Features Of Object-Oriented Programming

- Inheritance.
- Encapsulation.
- Polymorphism and Overloading.
- Data Hiding.
- Reusability.

Feature: Inheritance

- Allows classes to inherit features from other classes.
- Creates a parent-child relationship between classes.
- Promotes code reuse and hierarchy.

Feature: Encapsulation

Bundles related data and methods together.

Controls access to internal details.

Provides a clean interface for using objects.

Feature: Polymorphism and Overloading

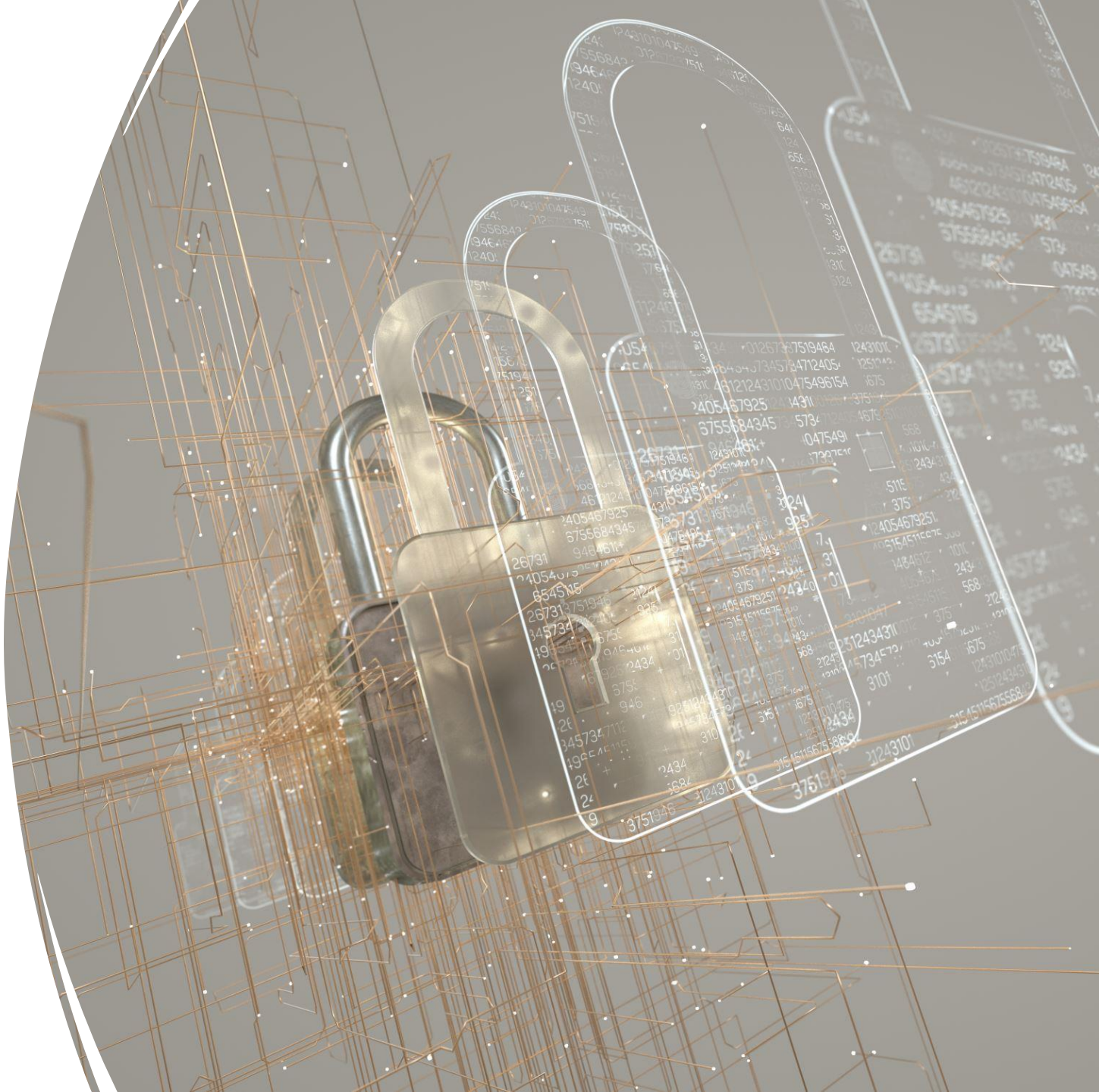
Objects can take different forms while sharing interface.

Methods can have different implementations.

Allows for flexible and extensible code.

Feature: Data Hiding

- Restricts direct access to object data.
- Uses private and protected attributes.
- Prevents unauthorized modifications.



Feature: Reusability

Write	Share	Reduce
Write once, use many times (FIFA or EA FC).	Share code across projects.	Reduce duplication.

Real-World Applications

Complex software systems.

Game development.

GUI applications.

Web applications.

Enterprise software.

Mobile app development.

Advantages of OOP


Better organization of code.


Easier maintenance.


Code reusability.


Scalability.


Security through data hiding.


Natural modeling of real-world entities.

Disadvantages of OOP

- Complexity: OOP can be harder to learn.
- Overhead: Can be slower sometimes.
- Design: Good OOP design is tough.
- Time: Initial development can take longer.
- Not Always Best: Overkill for simple tasks.
- Coupling: Classes can become too interconnected.
- Abstraction: Can be overused and confusing.

Next Time

Event Driven Programming