OOP_Intro

Thursday, August 19, 2021 10:18 PM

OO_Intro

Object Oriented Programming

CS F213

Tu Th Sa 10- 11 AM–Google Meet
and Th 4-6 PM- Quanta

CS F213

- Prof. Neena Goveas (IC)
- Prof Basabdatta Sen Bhattacharya, Dr. Swati Agarwal, Dr. Sujith Thomas, Dr. Swaroop Ravindra Joshi
- Ankush Soni and Farha Masroor
- FDTA

CS F213

- Lectures- Google Meet
- Lecture material- Google Classroom
- Lab- Quanta (begins- Aug 26th)
- Lab Quiz –Quanta (begins- Aug 26th)

Evaluations

Component	Duration	Date	% Weightage	Remarks
Lab quiz	Continuous.		20%	Open Book
Lab	Continuous		10%	Open Book
Programming test		9th Oct and 20th Nov	20%	Open Book
Group Project	Continuous		20%	Open Book
End-term exam		Dec 15 AN	30%	Open/Closed Book

→ No Mid test

Object Oriented Programming

- You have learnt – Procedural programming
- Here we learn a new approach

Procedures – Methods/functions
↳ way of organizing code

Adv :
organized
call multiple times
f1 – library.
↳ no errors
↳ tested

other code

main {
f1
f2
f1
}

f1 {
}

f2 {
}

Programming progression…

- Programming has progressed through:
  - machine code
  - assembly language
  - machine-independent programming languages
  - procedures & functions
  - objects

001
ADD . . .
} hardware

→ organizing

↳

Procedural Approach

- Focus is on procedures
- All data is shared; no protection
- More difficult to modify
- Hard to manage complexity

Array1
1 → 1D ~~Name~~ Gpa
2
3

Data
marks[1][]
1 19
7
16
12
19

Avg {marks[1][]}
{
return Aval
}

- More difficult to modify
- Hard to manage complexity

2
3
4
5

Yash
CGPA

12
19
20
Max

Not be allowed

Team T1 funT1a() { }
T2
T3

Data → Associated with real objects
marks
student
Questions.

Simulation, Games
Events.
car1 ← move/change
car2
car3
Yash

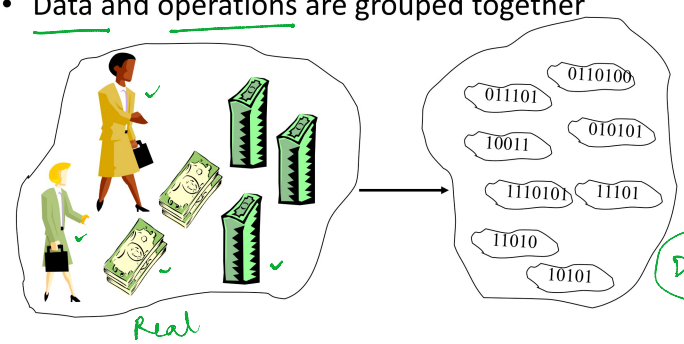
How do you handle events.

History of Object-Oriented Programming

- Started out for simulation of complex man-machine systems, but was soon realized that it was suitable for all complex programming projects
- SIMULA I (1962-65) and Simula 67 (1967) were the first two object-oriented languages
  - Developed at the Norwegian Computing Center, Oslo, Norway by Ole-Johan Dahl and Kristen Nygaard

OOP: Mapping the world to software

- Objects in the problem domain are mapped to objects in software
- Data and operations are grouped together



Object.

Data + operations

Static Array [N][M] x 1000 1000

student.

Flexibilities how many → properties → functions

Student + student ops → only

Who can use

IB No | Name | CGPA | Coursework

Old Paradigm

Computer system consists of data and programs.

Programs manipulate data.

Programs organized by

- functional decomposition
- dataflow
- modules

New Paradigm

Computer system consists of a set of objects.

Objects are responsible for knowing and doing certain things.

data functions.

Objects collaborate to carry out their

Procedural prog

car1 car2

o1 o2 o3

d1 f1 d2 f2

Responsibility

Objects are responsible for knowing and doing certain things.

data functions.

Objects collaborate to carry out their responsibilities.

Programs organized by classes, inheritance hierarchies and subsystems

Responsibility

Goal

Course

Goal:
Reach end line fastest.

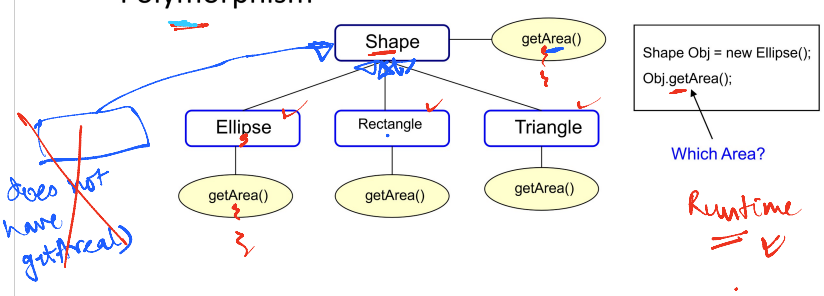
Encapsulation

data + methods

permissions

The 3 Principles of Object Oriented Programming

- Encapsulation
  - Control access to the data and methods
- Inheritance
  - Inherit the data and methods of parent class
- Polymorphism



Hierarchies

Parent

Child 1

Child 2

Shape

class

Any class which is a Child class of shape

Ellipse is a Shape

also have getArea()

Agenda

Object
Object Collaboration
Message
Class

OO Basics - Object

What Is an Object ?

OO Basics - Object

Real-world objects share two characteristics :-

They all have **state** and **behavior**.

For example,
dogs have state (name, color, breed, hungry)
and behavior (barking, fetching, wagging tail).

Bicycles have state (current gear, current pedal cadence, two wheels, number of gears)
and behavior (braking, accelerating, slowing down, changing gears).

Abstract
your requirement

OO Basics - Object

Software objects are modeled after real-world objects in that they too have state and behavior.

A software object maintains its state in one or more variables.

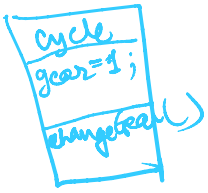
A variable is an item of data named by an identifier.

A software object implements its behavior with methods.

A method is a function (subroutine) associated with an object.

{1 2 3 4 5}
gear
int gear; → state

simpler
~~gear~~



Abstract.

OO Basics - Object

You can represent real-world objects by using software objects.

You might want to represent a real-world bicycle as a software object in the program that controls an electronic exercise bike.

You can also use software objects to model abstract concepts.

For example, an *event* is a common object used in window systems to represent the action of a user pressing a mouse button or a key on the keyboard.

→ has a software objects. interface ↳ inputs.

object.

OO Basics - Object

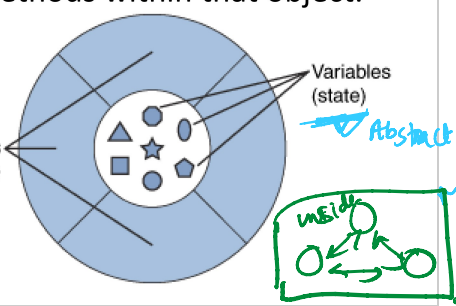
A software object.

Everything the software object knows (state) and can do (behavior) is expressed by the variables and the methods within that object.

procedural

```
main()
{ int c;
  input (5);
  changeGear(5);
}
```

infinite loop → breaks



→ Abstract

System

ElecBike

gear = int
distance = float
time = 0 float
⋮

methods → changeGear(int)

OO Basics - Object

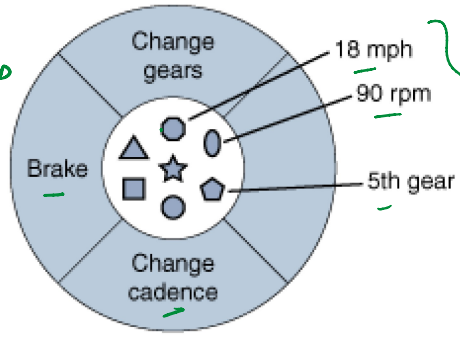
A software object that models your real-world bicycle would have variables that indicate the bicycle's current state :-

Its speed is 18 mph, its pedal cadence is 90 rpm, and its current gear is 5th.

These variables are formally known as instance variables because they contain the state for a particular bicycle object; in object-oriented terminology, a particular object is called an *instance*.

OO Basics - Object

The following figure illustrates a bicycle modeled as a software object.



OO Basics - Object

In addition to its variables, the software bicycle would also have methods to brake, change the pedal cadence, and change gears.

(It would not have a method for changing its speed because the bike's speed is just a side effect of which gear it's in and how fast the rider is pedaling.)

These methods are known formally as instance methods because they inspect or change the state of a particular bicycle instance.

OO Basics - Message

Software objects interact and communicate with each other using messages.

↳ Methods
↳ call

outside
– event

bl, br → crash

OO Basics - Message

What Is a Message?

A single object alone generally is not very useful.

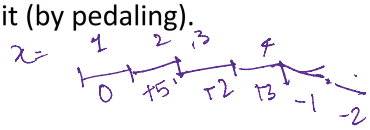
Object as a component of a larger program or application - contains many other objects.

Interaction of these objects achieve

- higher-order functionality
- complex behavior.

Your bicycle by itself, it is incapable of any activity;

the bicycle is useful only when another object (you) interacts with it (by pedaling).



procedural

Bike Race

Bike → position

B1 ↔ You

B2 x=0 resistance=0 resistance+3

B3 ↓ There

↳ random

B1

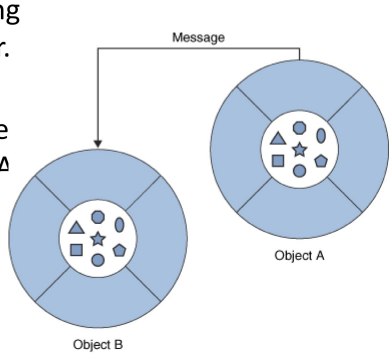
B2 . . .

event

OO Basics - Message

Software objects interact and communicate with each other by sending *messages* to each other.

When object A wants object B to perform one of B's methods, object A sends a message to object B



OO Basics - Message

Sometimes, the receiving object needs more information so that it knows exactly what to do :-

for example, when you want to change gears on your bicycle, you have to indicate which gear you want.

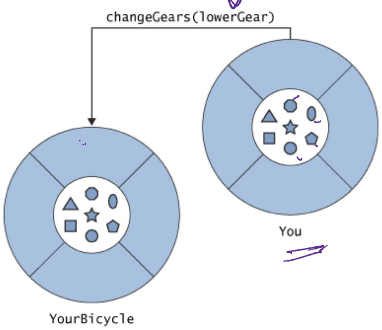
This information is passed along with the message as *parameters*.

variables. ID No - } Personal
Name - }

User → Method
Choose - Path
Change Gear } event
Exercise
bike
gears
rce Istyle
Changegear
Model

OO Basics - Message

parameter



Group 2

OO Basics - Message

The figure shows the three parts of a message :-

- The object to which the message is addressed (YourBicycle)
- The name of the method to perform (changeGears)
- Any parameters needed by the method (lowerGear)

OO Basics - Message

Messages provide two important benefits :-

- An object's behavior is expressed through its methods, so (aside from direct variable access) message passing supports all possible interactions between objects.
- Objects don't need to be in the same process or even on the same machine to send messages back and forth and receive messages from each other.

OO Basics - Class

A class is a blueprint or prototype that defines the variables and the methods common to all objects of a certain kind.

OO Basics - Class

What Is a Class?

In the real world, you often have many objects of the same kind.

For example, your bicycle is just one of many bicycles in the world.

Using object-oriented terminology, we say that your bicycle object is an instance of the class of objects known as bicycles.

OO Basics - Class

Bicycles have some state

(current gear, current cadence, two wheels)

and behavior (change gears, brake) in common.

However, each bicycle's state is independent of and can be different from that of other bicycles.

When building them, manufacturers take advantage of the fact that bicycles share characteristics, building many bicycles from the same blueprint. It would be very inefficient to produce a new blueprint for every bicycle manufactured.

OO Basics - Class

In object-oriented software, it's also possible to have many objects of the same kind that share characteristics like :- rectangles, employee records, video clips, and so on.

Like bicycle manufacturers, you can take advantage of the fact that objects of the same kind are similar and you can create a blueprint for those objects.

A software blueprint for objects is called a class.

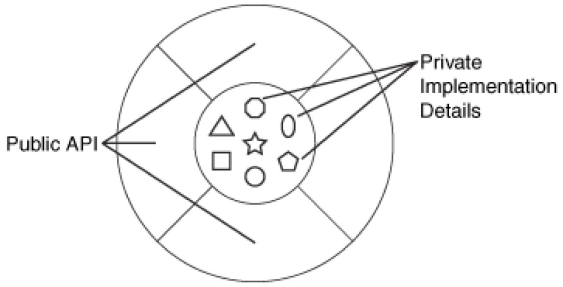
OO Basics - Class

The class for our bicycle example would declare the instance variables necessary to contain the current gear, the current cadence, and so on for each bicycle object.

The class would also declare and provide implementations for the instance methods that allow the rider to change gears, brake, and change the pedaling cadence.

OO Basics - Class

The bicycle class.



OO Basics - Class

After you've created the bicycle class, you can create any number of bicycle objects from that class.

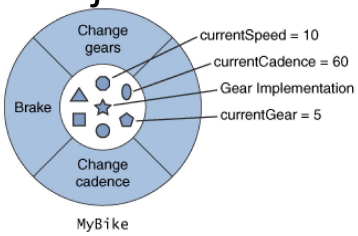
When you create an instance of a class, the system allocates enough memory for the object and all its instance variables.

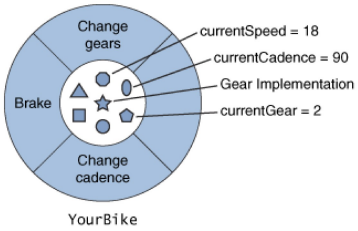
Each instance gets its own copy of all the instance variables defined in the class.

OO Basics - Object

MyBike and **YourBike** are two different instances of the Bike class.

Each instance has its own copy of the instance variables defined in the Bike class but has different values for these variables.





OO Basics - Variables

In addition to instance variables, classes can define class variables.

A class variable contains information that is shared by all instances of the class.

A class can also declare class methods .

You can invoke a class method directly from the class, whereas you must invoke instance methods on a particular instance.