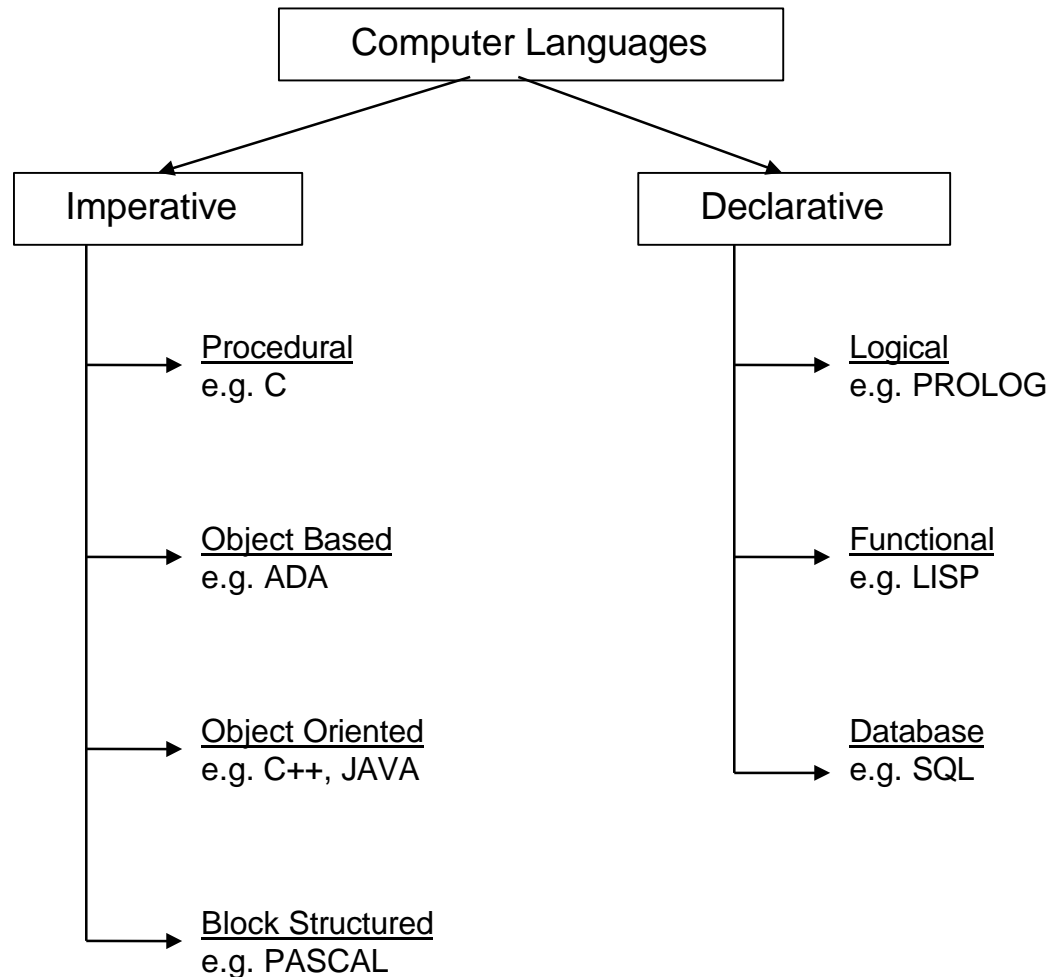


Dart – Object Oriented Programming

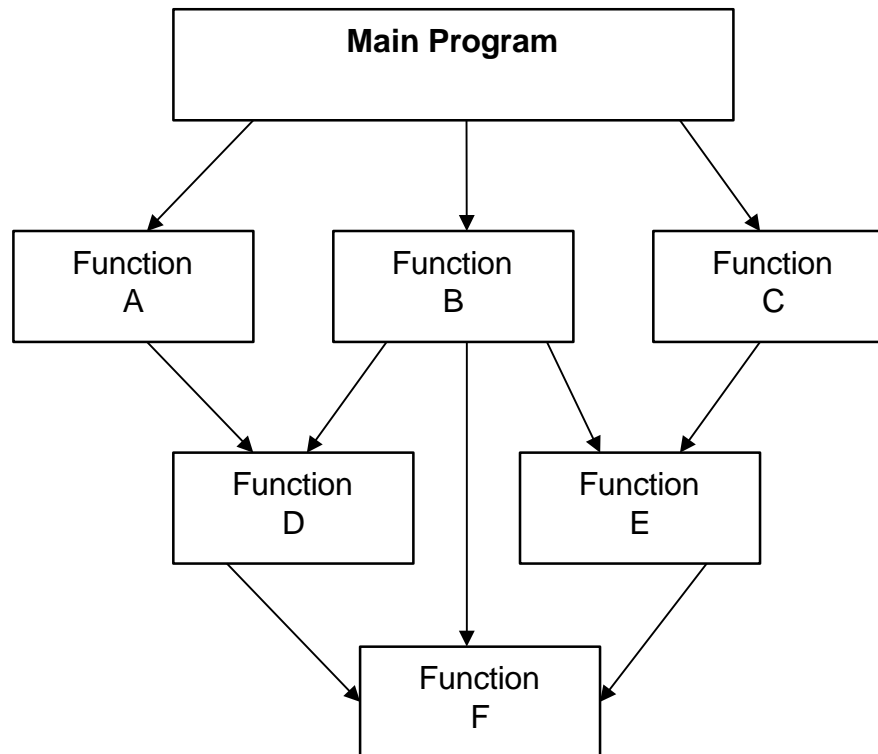
Girma B.



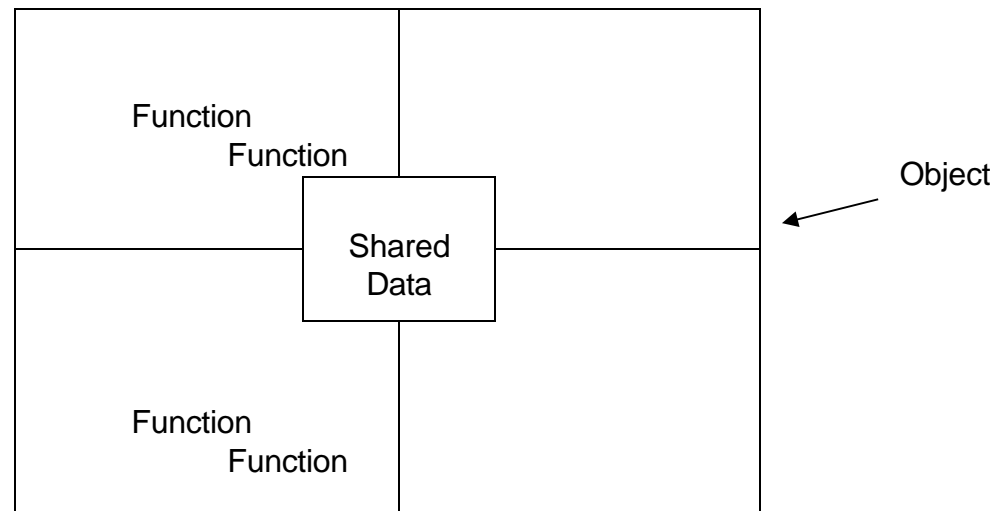
Programming Languages



Procedure Oriented Programming



Object Oriented Programming



What is Object Oriented Programming?

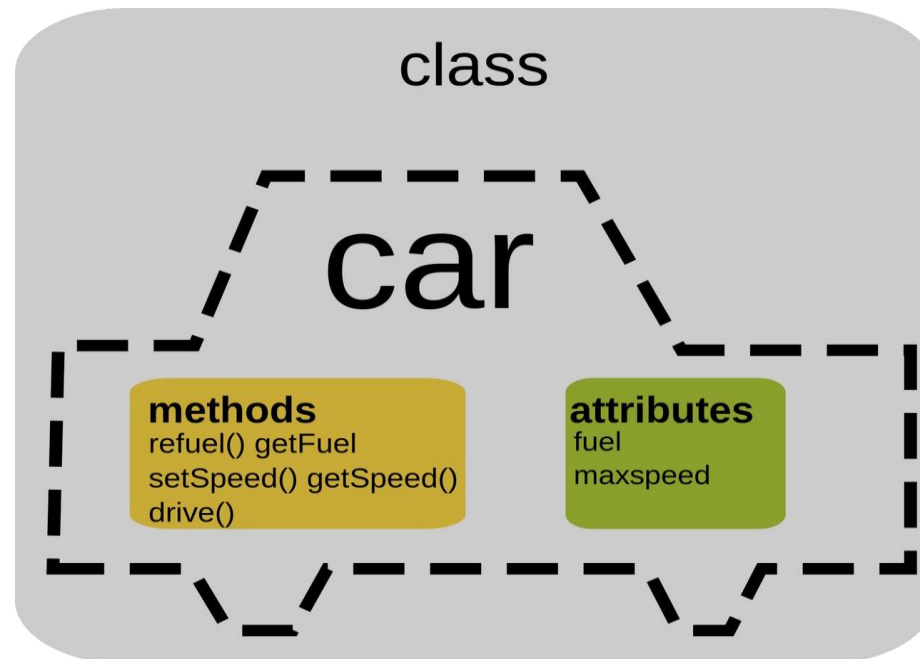
- Object-oriented Programming, or OOP for short, is a programming paradigm which provides a means of structuring programs so that properties and behaviors are bundled into individual objects.
- For instance, an object could represent a person with a name property, age, address, etc., with behaviors like walking, talking, breathing, and running. Or an email with properties like recipient list, subject, body, etc., and behaviors like adding attachments and sending.

What is Object Oriented Programming?

- Put another way, object-oriented programming is an approach for modeling concrete, real-world things like cars as well as relations between things like companies and employees, students and teachers, etc.
- OOP models real-world entities as software objects, which have some data associated with them and can perform certain functions.

Class

Classes are used to create new user-defined data structures that contain arbitrary information about something. In the case of an car, we could create an Car() class to track properties about the Car like the fuel and maxspeed.

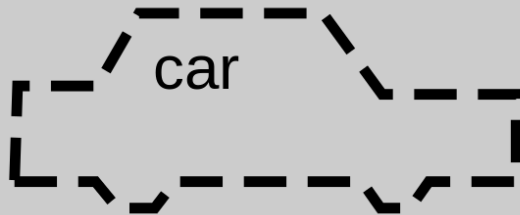


Objects

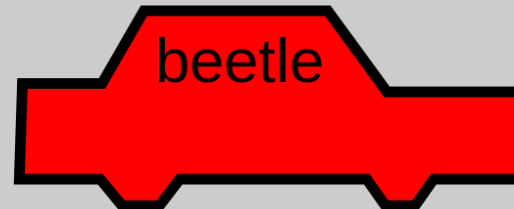
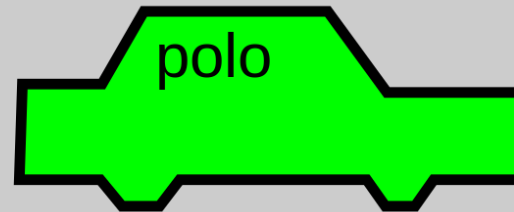
- While the class is the blueprint, an instance is a copy of the class with actual values, literally an object belonging to a specific class.
- Put another way, a class is like a form or questionnaire.
- It defines the needed information. After you fill out the form, your specific copy is an instance of the class; it contains actual information relevant to you.

Objects

class



objects



OOP Features

- Reduction in the complexity
- Importance of data
- Creation of new data structure
- Data Hiding
- Characterization of the objects
- Communication among objects
- Extensibility
- Bottom-up programming approach

OOP Concepts

- Class
- Object
- Data Hiding / abstraction / encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding

Creating a class

```
class ClassName {  
    <fields>  
    <getter/setter>  
    <constructor>  
    <functions>  
}
```

Constructor

- A constructor is a different type of function which is created with same name as its class name.
- The constructor is used to initialize an object when it is created. When we create the object of class, then constructor is automatically called.
- It is quite similar to class function but it has no explicit return type.
- The generative constructor is the most general form of the constructor, which is used to create a new instance of a class.

Constructor No Agruments

- A Constructor which has no parameter is called default constructor or no-arg constructor. It is automatically created (with no argument) by
- Dart compiler if we don't declare in the class. The Dart compiler ignores the default constructor if we create a constructor with argument or no argument.

Constructor Parameters

- We can also pass the parameters to a constructor that type of constructor is called parameterized constructor.
- It is used to initialize instance variables. Sometimes, we need a constructor which accepts single or multiple parameters.
- The parameterized constructors are mainly used to initialize instance variable with own values.

Constructor Named

- The named constructors are used to declare the multiple constructors in single class. The syntax is given below.
- Syntax:

`className.constructor_name(param_list)`

this

- The this keyword is used to refer the current class object. It indicates the current instance of the class, methods, or constructor.
- It can be also used to call the current class methods or constructors. It eliminates the uncertainty between class attributes and the parameter names are the same.
- If we declare the class attributes same as the parameter name, that situation will create ambiguity in the program, then the this keyword can remove the ambiguity by prefixing the class attributes.
- It can be passed as an argument in the class method or constructors.

Constructor

```
void main() {  
    // Creating an object  
    Student std = new Student("Alisha",26);  
}  
  
class Student{  
    // Declaring a construtor  
    Student(String str, int age){  
        print("The name is: ${str}");  
        print("The age is: ${age}");  
    }  
}
```

Function - why?

- It increases the module approach to solve the problems.
- It enhances the re-usability of the program.
- We can do the coupling of the programs.
- It optimizes the code.
- It makes debugging easier.
- It makes development easy and creates less complexity.

Function - Definition

- A function can be defined by providing the name of the function with the appropriate parameter and return type. A function contains a set of statements which are called function body. The syntax is given below.

- Syntax:

```
return_type func_name (parameter_list):  
{  
    //statement(s)  
    return value;  
}
```

Function - Example

```
int mul(int a, int b){  
    int c;  
    c = a+b;  
    print("The sum is:${c}");  
}
```

Function - Parameters

- When a function is called, it may have some information as per the function prototype is known as a parameter (argument).
- The number of parameters passed and data type while the function call must be matched with the number of parameters during function declaration. Otherwise, it will throw an error.
- Parameter passing is also optional, which means it is not compulsory to pass during function declaration. The parameter can be two types.
- Actual Parameter - A parameter which is passed during a function definition is called the actual parameter.
- Formal Parameter - A parameter which is passed during a function call is called the formal parameter.

Function - Parameters


- When a function is called, it may have some information as per the function prototype is known as a parameter (argument).
- The number of parameters passed and data type while the function call must be matched with the number of parameters during function declaration. Otherwise, it will throw an error.
- Parameter passing is also optional, which means it is not compulsory to pass during function declaration. The parameter can be two types.
- Actual Parameter - A parameter which is passed during a function definition is called the actual parameter.
- Formal Parameter - A parameter which is passed during a function call is called the formal parameter.

Function - Parameters

- A function always returns some value as a result to the point where it is called. The return keyword is used to return a value.
- The return statement is optional. A function can have only one return statement. The syntax is given below.

```
int sum (int a, int b){  
    .....  
    .....  
    return result;  
}
```

```
var c = sum(30,20) ←
```



Function - Anonymous

- Dart also provides the facility to specify a nameless function or function without a name. This type of function is known as an anonymous function, lambda, or closure.
- An anonymous function behaves the same as a regular function, but it does not have a name with it. It can have zero or any number of arguments with an optional type annotation.
- We can assign the anonymous function to a variable, and then we can retrieve or access the value of the closure based on our requirement.

Function - Anonymous

- An Anonymous function contains an independent block of the code, and that can be passed around in our code as function parameters. The syntax is as follows.

Syntax:

```
(parameter_list) {  
    statement(s)  
}
```

Function - Anonymous

```
void main() {  
    var list = ["Ram","Raj","Rani","Ritu"];  
    print("Example of anonymous function");  
    list.forEach((item) {  
        print('${list.indexOf(item)}: $item');  
    });  
}
```

Thank you