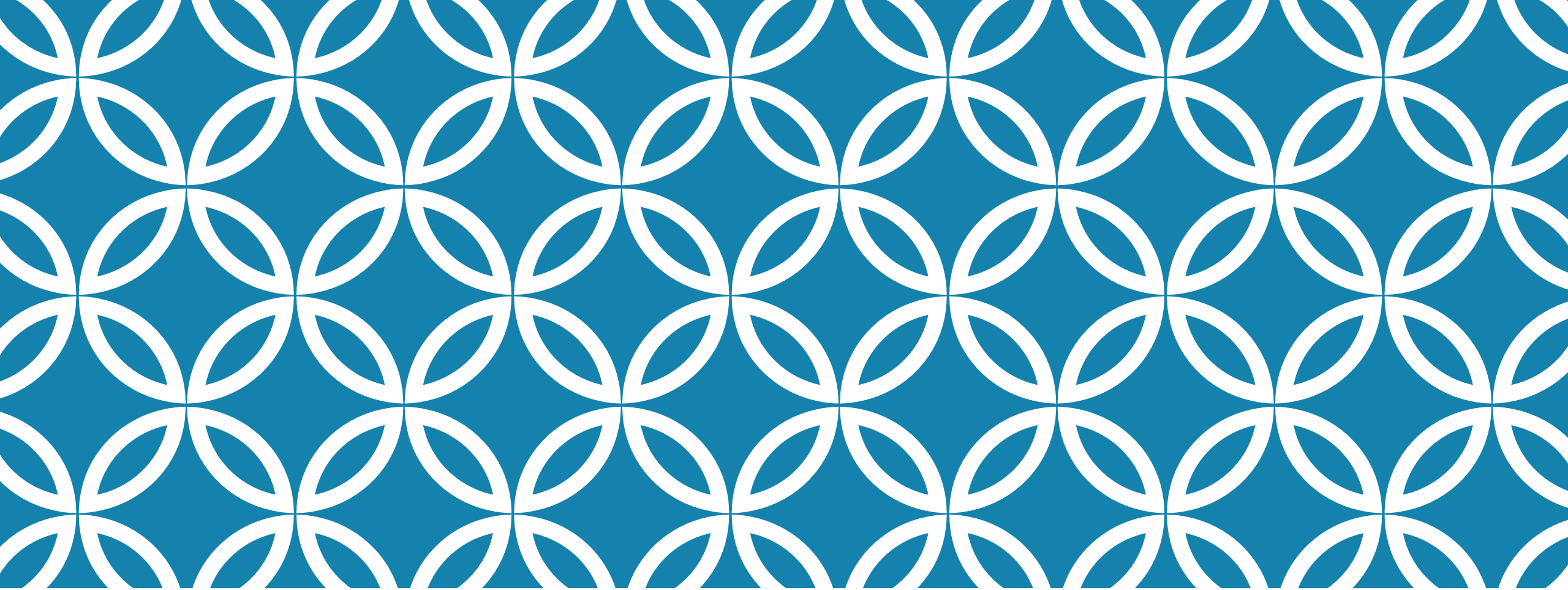


OBJECT ORIENTED PROGRAMMING

Abdullah Bin Kasem Bhuiyan

Toptic 1: Introduction

WHAT IS AN OBJECT?

In its basic definition, an object is an entity that contains both data and behavior.

- This is the key difference between the more traditional programming methodology, procedural programming, and O-O programming.

PROCEDURAL PROGRAMMING

In procedural programming, code is placed into methods.

- Ideally these procedures then become "black boxes", inputs come in and outputs go out.
  - Data is placed into separate structures, and is manipulated by these methods.

O-O PROGRAMMING

The fundamental advantage of O-O programming is that the data and the operations that manipulate the data are both contained in the object.

- For example, when an object is transported across a network, the entire object, including the data and behavior, goes with it.

OBJECT DATA

The data stored within an object represents the *state* of the object.

- In O-O programming terminology, this data is called *attributes*.

OBJECT BEHAVIORS

The *behavior* of an object is what the object can do.

- In procedural languages the behavior is defined by procedures, functions, and subroutines.

WHAT EXACTLY IS A CLASS?

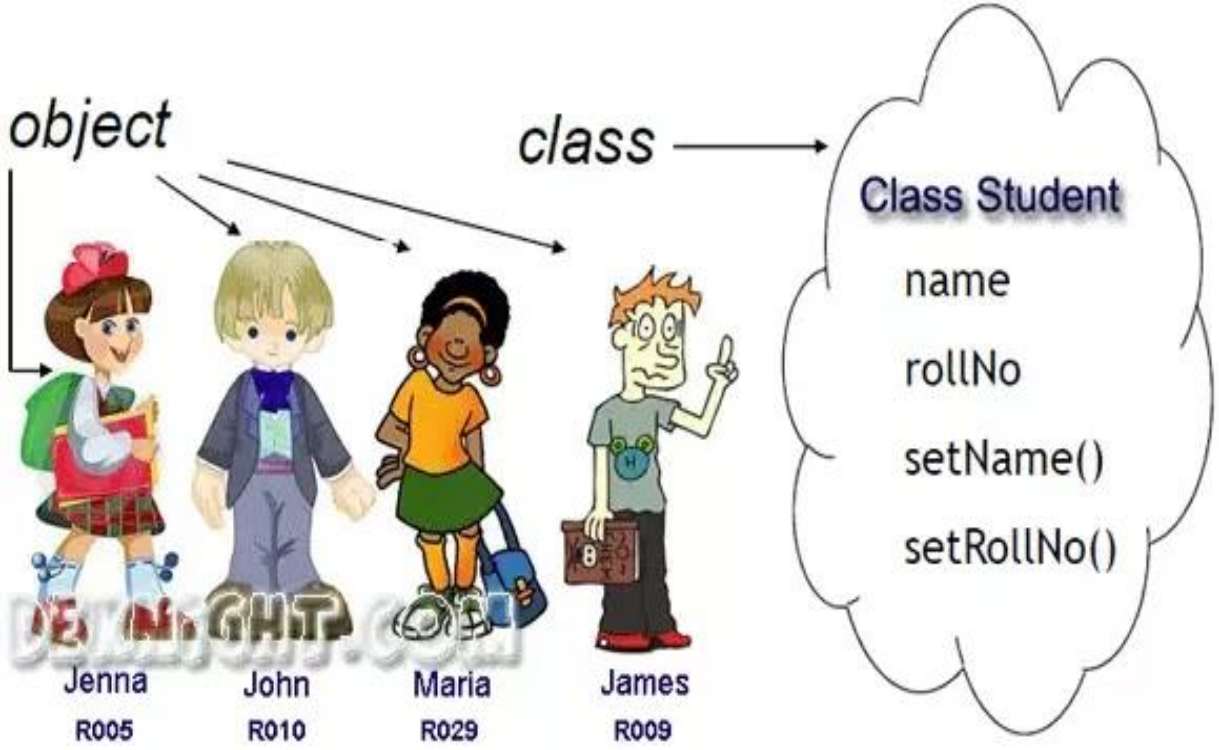
A *class* is a blueprint/template for an object.

- When you instantiate an object, you use a class as the basis for how the object is built.

WHAT EXACTLY IS A CLASS?

An object cannot be instantiated without a class.

- Classes can be thought of as the templates, or cookie cutters, for objects as seen in the next figure.



PROCEDURAL VS OBJECT ORIENTED

Procedural Oriented Programming

- In procedural programming, program is divided into small parts called ***functions***.
- Procedural programming follows ***top down approach***.
- There is no access specifier in procedural programming.
- Procedural programming does not have any proper way for hiding data so it is ***less secure***.

Object Oriented Programming

- In object oriented programming, program is divided into small parts called ***objects***.
- Object oriented programming follows ***bottom up approach***.
- Object oriented programming have access specifiers like private, public, protected etc.
- Object oriented programming provides data hiding so it is ***more secure***.

PROCEDURAL VS OBJECT ORIENTED

Procedural Oriented Programming

- In procedural programming, overloading is not possible.
- Procedural programming is based on ***unreal world***.
- Examples: C, FORTRAN, Pascal, Basic etc.

Object Oriented Programming

- Overloading is possible in object oriented programming.
- Object oriented programming is based on ***real world***.
- Example: C++, Java, Python, C# etc.

BENEFITS OF OBJECT ORIENTATION

- ✓Faster development,
- ✓Reusability,
- ✓Increased quality
- ✓Information Hiding
- ✓Easy debugging
- ✓Object technology emphasizes modeling the real world and provides us with the stronger equivalence of the real world‘s entities (objects) than other methodologies.
- ✓Raising the level of abstraction to the point where application can be implemented in the same terms as they are described.

WHY OBJECT ORIENTATION?

To create sets of objects that work together concurrently to produce s/w that better, model their problem domain that similarly system produced by traditional techniques.

- ✓It adapts to
  - ❑Changing requirements
  - ❑Easier to maintain
  - ❑More robust
  - ❑Promote greater design
  - ❑Code reuse
- ✓Higher level of abstraction
- ✓Seamless transition among different phases of software development
- ✓Encouragement of good programming techniques
- ✓Promotion of reusability

A CASE STUDY - A PAYROLL PROGRAM

Consider a payroll program that processes employee records at a small manufacturing firm. This company has three types of employees:

- ✓Managers: Receive a regular salary.
- ✓Office Workers: Receive an hourly wage and are eligible for overtime after 40 hours.
- ✓Production Workers: Are paid according to a piece rate.

STRUCTURED APPROACH

```
FOR EVERY EMPLOYEE DO
BEGIN

    IF employee = manager THEN

        CALL computeManagerSalary

    IF employee = office worker THEN

        CALL computeOfficeWorkerSalary


    IF employee = production worker THEN

        CALL computeProductionWorkerSalary


END
```

STRUCTURED APPROACH

```
FOR EVERY EMPLOYEE
BEGIN
     IF employee = manager THEN
          CALL computeManagerSalary
     IF employee = office worker THEN
          CALL computeOfficeWorker_salary
     IF employee = production worker THEN
          CALL computeProductionWorker_salary
     IF employee = temporary office worker THEN
          CALL computeTemporaryOfficeWorkerSalary
     IF employee = junior production worker THEN
          CALL computeJuniorProductionWorkerSalary
END
```

AN OBJECT-ORIENTED APPROACH

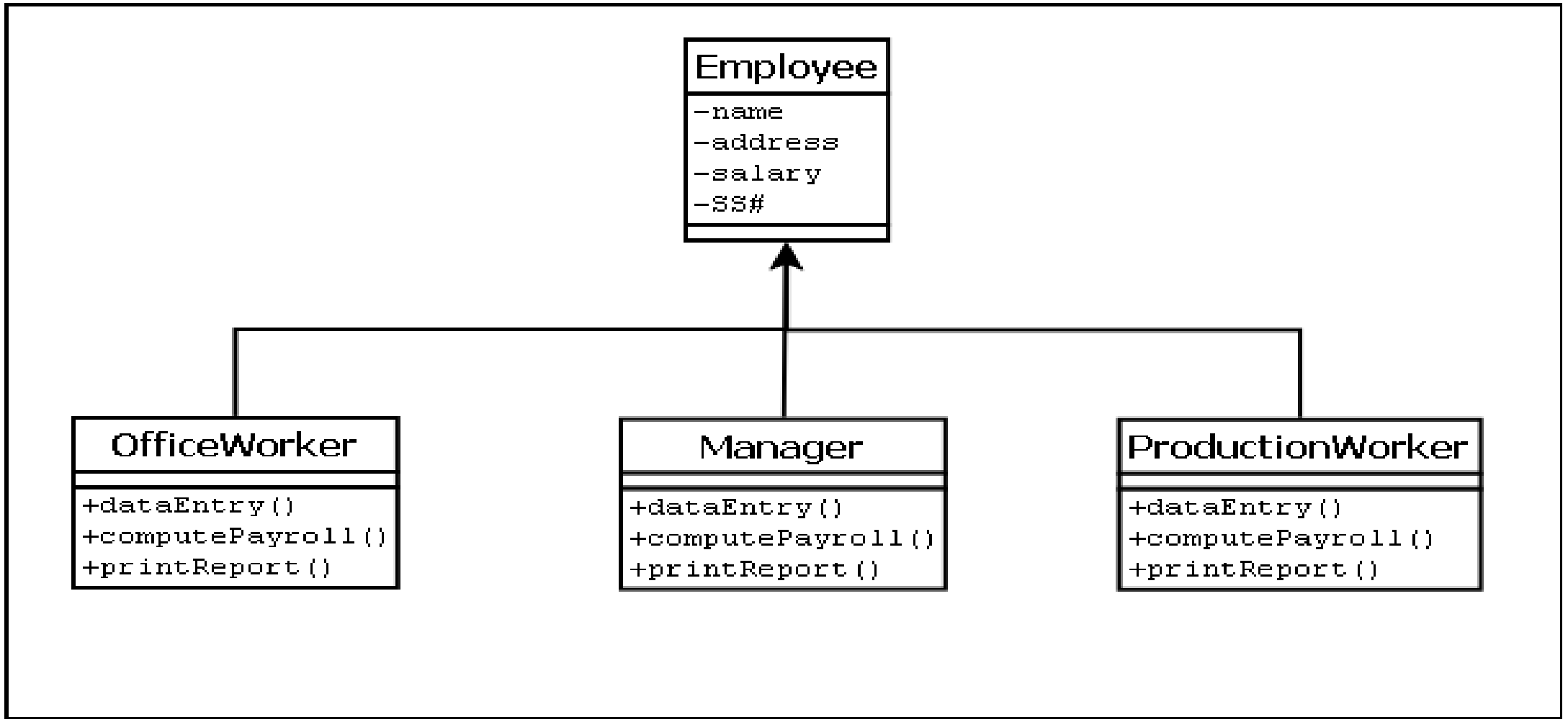
What objects does the application need?
The goal of OO analysis is to identify objects and classes that support the problem domain and system's requirements.
Some general candidate classes are:

- Persons
- Places
- Things

Class Hierarchy

- Identify class hierarchy
- Identify commonality among the classes
- Draw the general-specific class hierarchy.

AN OBJECT-ORIENTED APPROACH



OO APPROACH

```
FOR EVERY EMPLOYEE DO
BEGIN
    employee computePayroll
END
```

MAIN CONCEPTS OF OOP

1. **Encapsulation---**binding data and operations together
2. **Abstraction----**hiding the implementation complexity
3. **Polymorphism---**many forms of an object
4. **Inheritance---**taking over the common attributes and operation from a class.