Dr. D. Y. Patil Pratishthan's
Institute for Advanced Computing and Software Development

Day2

Sub-c++

Reference Variable

- A reference variable is an alias, that is, another name for an already existing variable.
- Once a reference is initialized with a variable, either the variable name or the reference name may be used to refer to the variable.
- The main use to support pass-by-reference.
- In an reference variable,it is passed into a function, the function works on the original copy (instead of a clone copy in pass-by-value).
- Changes inside the function are reflected outside the function.

- A reference is similar to a pointer. In many cases, a reference can be used as an alternative to pointer, in particular, for the function parameter.

- Difference between a pointer and reference-

Pointers	Reference
It's a separate variable that stores an address of another variable	It's an alternative name given to the variable
It has its own separate block of memory	It doesn't have a separate block of memory
It's a flexible connection i.e. a pointer declared can point to any variable, provided it's a non-const	It's a rigid connection. i.e. A reference associated with a variable while initialization can't be assigned to another variable.
It needs an indirection operator for dereferencing	It doesn't need any operator for dereferencing

Using a Reference

- While using references you should know …
  - References have to be initialized.
  - No memory is allocated to references.

Reference: Pass by Reference

- From the function call one cannot make out whether the parameters have been “passed by value” or “passed by reference”.

```
void swapRef(int
  &ref1,
               int
  &ref2)
{
  int temp;
  temp = ref1;
  ref1 = ref2;
  ref2 = temp;
}
```

```
int main()
{
  int n1=10, n2=20;
  //pass by reference
  swapRef(n1, n2);
  cout<<“n1=”<<n1;
  cout<<“n2=”<<n2;
  return 0;
}
```

function

- A function is a group of statements that together perform a task.
- Every C++ program has at least one function, which is main(), and all the most trivial programs can define additional functions.
- A function declaration tells the compiler about a function's name, return type, and parameters.
- A function definition provides the actual body of the function.
- The general form of a C++ function definition is as follows –

```
return_type function_name( parameter list ) {
   body of the function
}
```

Types of functions

- call by value
- call by address/pass by address (using pointer)
- call by address/pass by reference (using reference variable)
- default argument/parameter in function
- inline function
- friend function

Call by value

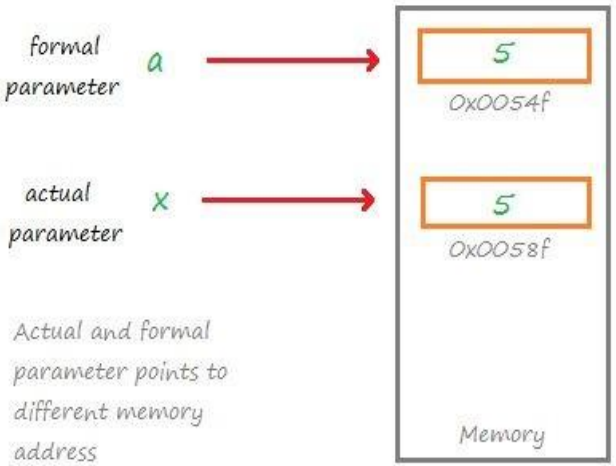
- widely used method.
- you don’t want your original values of the variables to be changed.
- only the values of the variables are passed.
- achieved by creating dummy variables in memory.

```
void main(){
    int x=8, y=6;
     int s=add(x,y);        //call by value
    cout<<s;
}
int add(int a, int b){
    return a+b
}
```


int x = 10;
address 0x64
int a = 10;
address 0x66

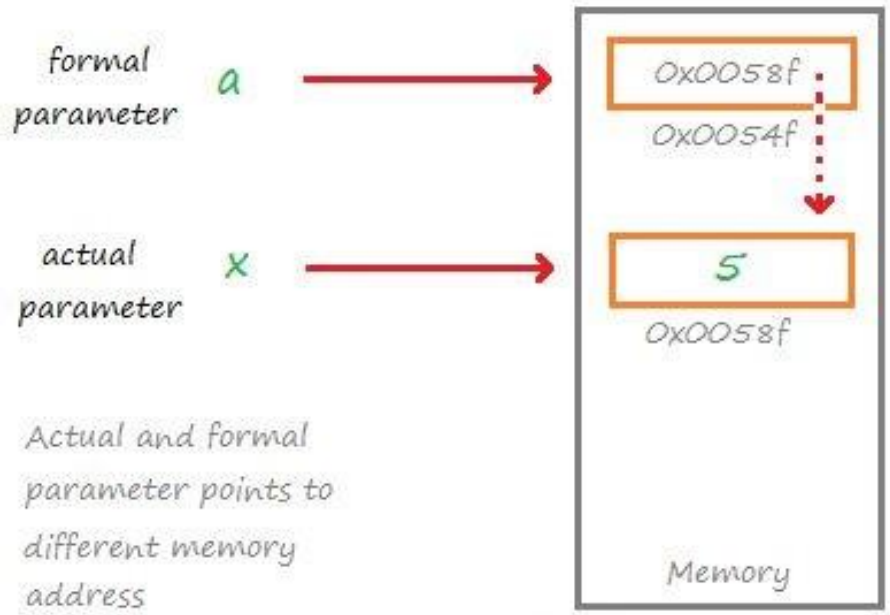
add(x, y)


int y = 20;
address 0x65
int b = 20;
address 0x67


formal parameter
a
actual parameter
x
5
0x0054f
5
0x0058f
Actual and formal parameter points to different memory address
Memory

call by address/pass by address

- The function arguments are passed as address.
- The caller function passes the address of the parameters.
- Pointer variables are used in the function definition.
- With the help of the Call by address method, the function can access the actual parameters and modify them.
- pointer variable holds the address of the actual parameter, hence the changes done by the formal parameter is also reflected in the actual parameter.

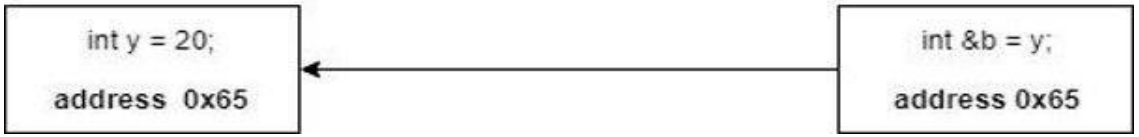

formal parameter
a
actual parameter
x
0x0058f
0x0054f
5
0x0058f
Memory
Actual and formal parameter points to different memory address

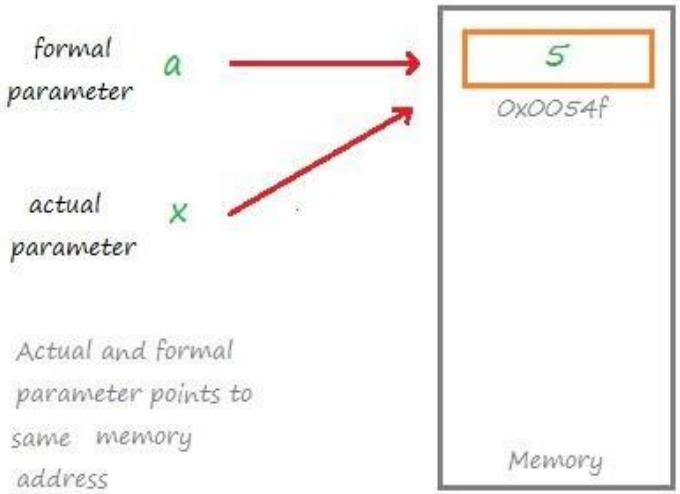
Call by reference / Pass by reference

- Dummy variables are not created,
- A reference of an already existing variable is passed to the method.
- This reference points to the same memory location
- Hence separate copies are not made in the memory.
- The important point to note - changes made in the reference variables are reflected in the actual variable.


int x = 10;
address 0x64
int &a = x;
address 0x64

add(x, y)


int y = 20;
address 0x65
int &b = y;
address 0x65


formal parameter
a
actual parameter
x
5
0x0054f
Memory
Actual and formal parameter points to same memory address

Default argument in function

-A default argument is a value provided in a function declaration that is automatically assigned by the compiler if the calling function doesn't provide a value for the argument.
-In case any value is passed, the default value is overridden.

```
int sum(int x, int y, int z = 0, int w = 0)  //assigning default values to z,w as 0
{
   return (x + y + z + w);
}
int main()
{
   cout << sum(10, 15) << endl;                  //25
  cout << sum(10, 15, 25) << endl;                //50
  cout << sum(10, 15, 25, 30) << endl;           //80
   return 0;
}
```

Inline Function

- C++ provides an inline functions to reduce the function call overhead.
- Inline function is a function that is expanded in line when it is called.
- When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call.
- This substitution is performed by the C++ compiler at compile time.
- Inline function may increase efficiency if it is small.
- inline functions to reduce the function call overhead.

```
inline int cube(int s)
{
   return s*s*s;
}
int main()
{    cout << "The cube of 3 is: " << cube(3) ;
} //Output: The cube of 3 is: 27
```

Remember,

inlining is only a request to the compiler, not a command.

Compiler can ignore the request for inlining.

Compiler may not perform inlining in such circumstances like:

1) If a function contains a loop. (for, while, do-while)

2) If a function contains static variables.

3) If a function is recursive.

4)If a function return type is other than void, and the return statement doesn’t exist in function body.

5) If a function contains switch or goto statement.

Inline functions provide following advantages:

1) Function call overhead doesn’t occur.

2)saves the overhead of push/pop variables on the stack when function is called.

3) It also saves overhead of a return call from a function.

C++ OOP Concepts

- The major purpose of C++ programming is to introduce the concept of object orientation to the C programming language.
- Object Oriented Programming is a paradigm that provides many concepts such as inheritance, data binding, polymorphism etc.
- The programming paradigm where everything is represented as an object is known as truly object-oriented programming language.
- Smalltalk is considered as the first truly object-oriented programming language.

OOPs (Object Oriented Programming System)

- **Object** means a real word entity such as pen, chair, table,fan etc.
- **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.
- It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

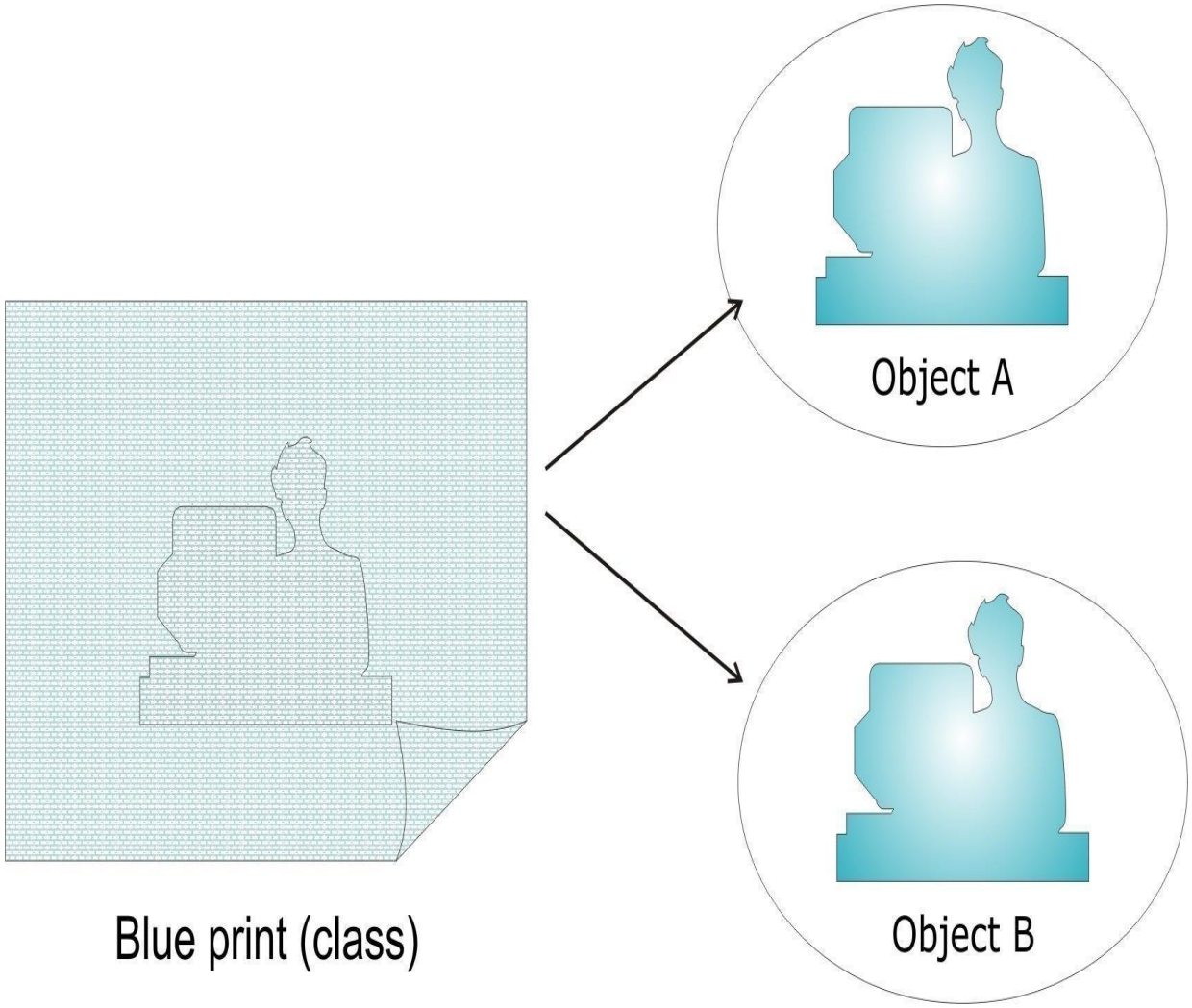
Class

- `struct` keyword can be replaced by `class` keyword.
  - C++ supports `struct` keyword for compatibility.
- Generally `struct` is used in ‘C’ context while `class` is used in C++ context.
- Class and object is C++ terminology.

Class

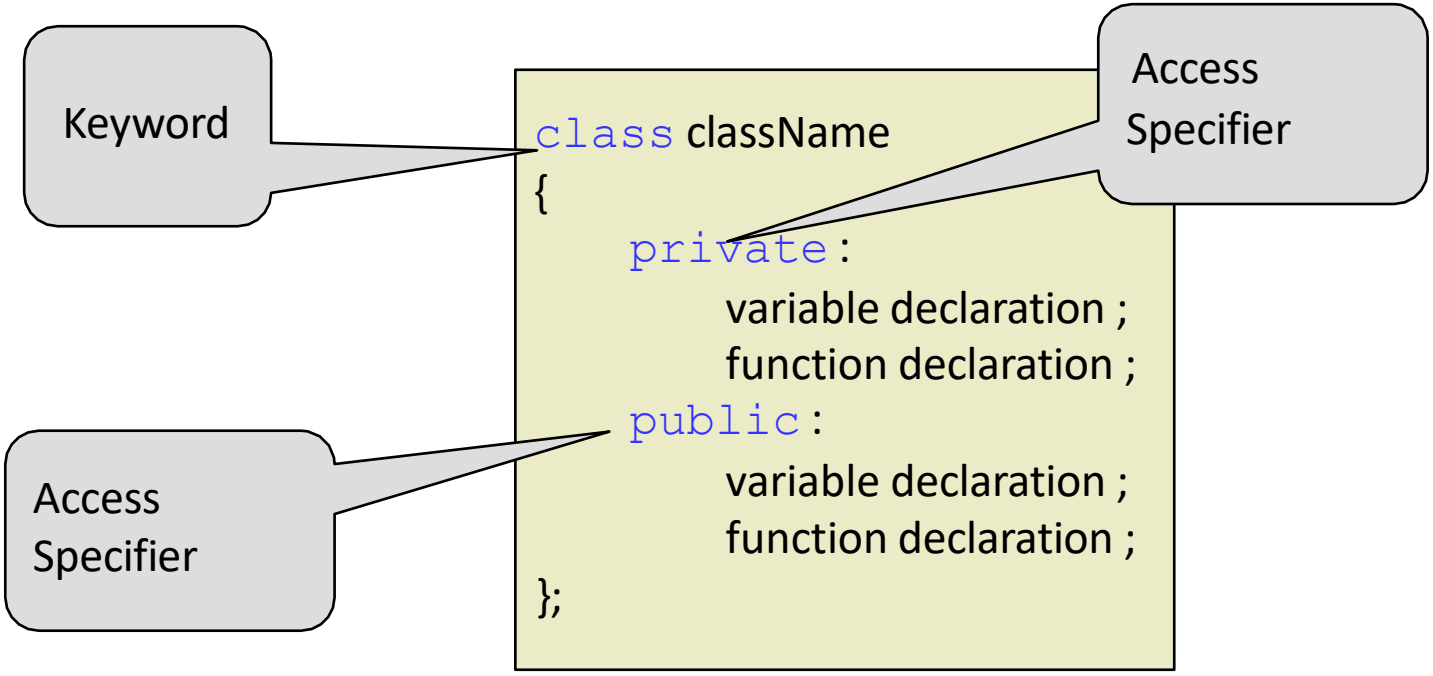
- A template for creating similar objects.
- Maps real world entities into classes through data members and member functions.
- A user defined type.
- An object is an instance of a class.
- By writing a class and creating objects of that class, one can map two concepts of object model, abstraction and encapsulation, into software domain.

Objects and Classes



Class

- Template for the creation of similar objects.
- A class in C++ is an encapsulation of data members and member functions that manipulate the data.



If semicolon is missing, compiler throws an error

- syntax error : missing ';' before 'PCH creation point' Error executing `cl.exe`

Class Components

- A class declaration consists of following components
  - Access specifiers: restrict access of class members
    - `private`
    - `protected`
    - `public`
  - Data members
  - Member functions
    - Constructors
    - Destructors
    - Normal Function

Syntax:

```
class class_name
{
access_specifier :
data_member declaration;
member_function
definition;
};
```

Three core components of class

1. **Access Specifier**
2. **Data Members**
3. **Member Functions**

Example:

```
class Employee
{
private:
        int empid;
        char * name;
        …
public:
        printdetails(){…}
        computesalary(){…
        }
        …
};
```

When defining a class there are important components as follows:

1. Access Specifier:

   It is a keyword that specifies the scope of the component. There are three specifiers

public	Accessible within class and outside class.
private	Accessible only within class.
protected	Accessible within class and its next derived class.

2. Data Members:

   It is definition of data/attributes of corresponding object. To achieve encapsulation the data is made private.

3. Member functions:

   It is the behavior/action/operations of an object. To achieve Encapsulation some functions are made public. There are three types of functions,

   a.Constructor b.Destructor c.Ordinary functions.

Object:

Object is an instance of a class. The process of creating objects is called as instantiation. When an object is created following happens:

a. Memory is allocated
b. Constructor is called.
c. Memory is initialized.

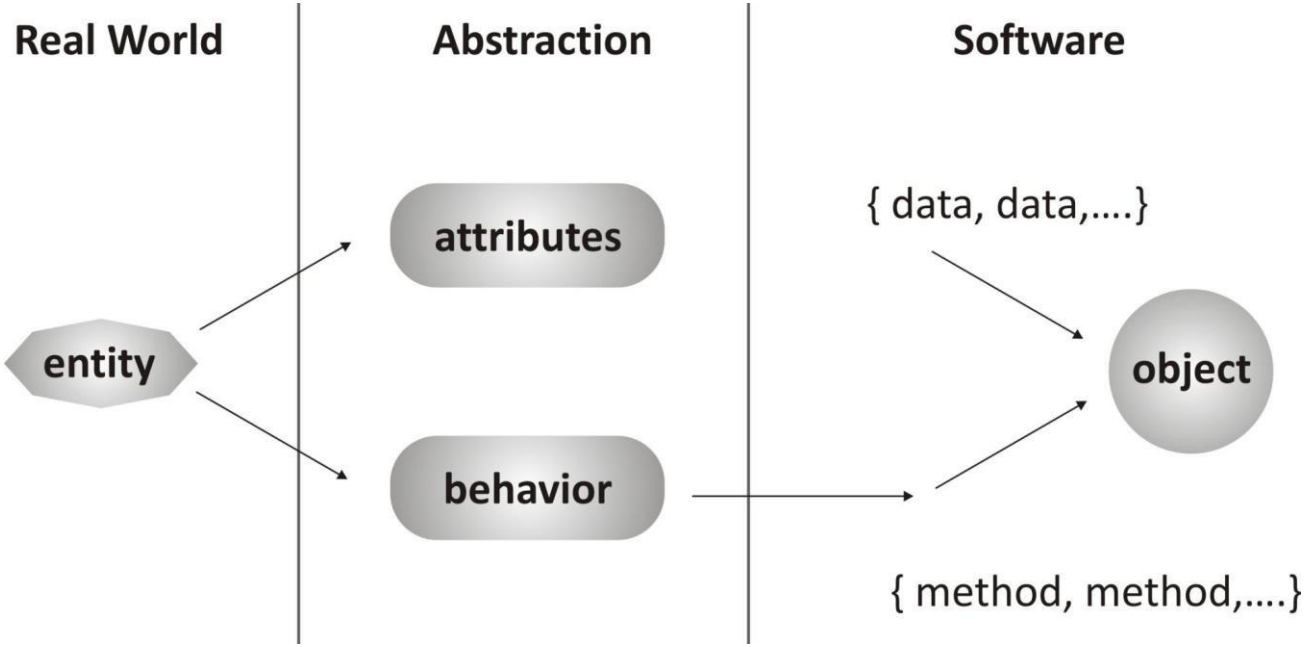
Abstraction

- Abstraction is the process of identifying the key aspects of an entity and ignoring the rest.
- Only those aspects are selected that are important to the current problem scenario.
- Example : Abstraction of a person object
  - Enumerate attributes of a “person object” that need to be created for developing a database
    - useful for social survey
    - useful for health care industry
    - useful for payroll system

Abstraction of a `Person` Object

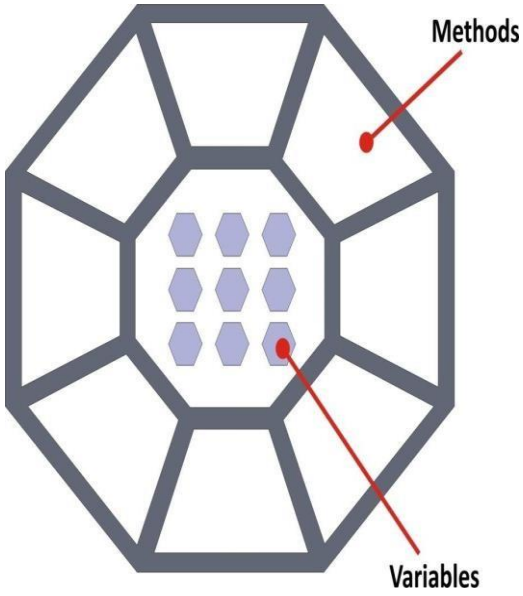
Social Survey	Health Care	Payroll System
name	name	name
Age	age	age
marital status	---	---
religion	---	---
income group	---	---
address	address	address
occupation	occupation	occupation
---	blood group	---
---	weight	---
---	previous record	---
---	---	basic salary
---	---	department
---	---	qualification

Abstraction



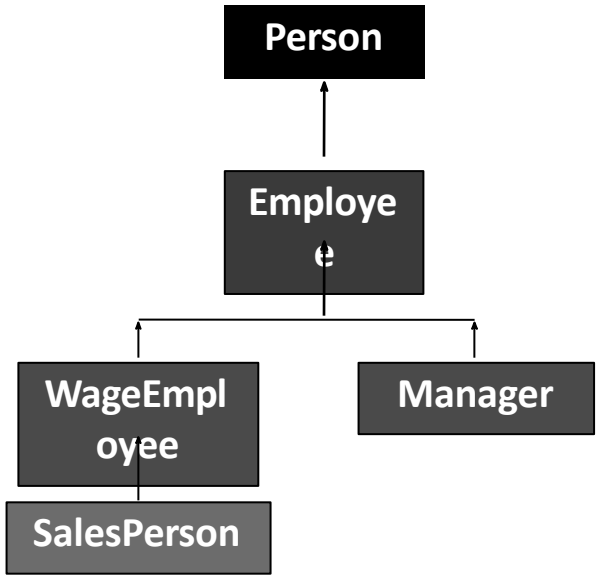
Encapsulation

- Encapsulation is a mechanism used to hide the data, internal structure, and implementation details of an object.
- All interaction with the object is through a public interface of operations.
- The user knows only about the interface; any changes to the implementation does not affect the user.



Inheritance

- Classification helps in handling complexity.
- Inheritance is the process by which one object can acquire the properties of another object.
  - Broad category is formed and then sub-categories are formed.
- “is – a” a kind of hierarchy.

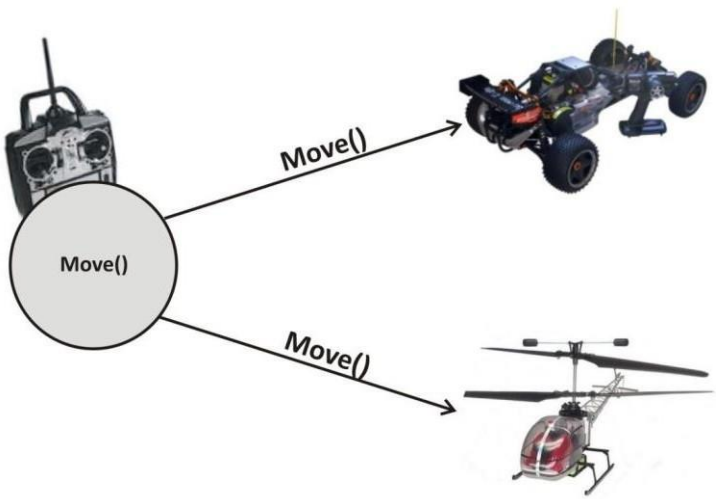


Inheritance

- Generalization
  - Factoring out common elements within a set of categories into a more general category called super-class.
  - Requires good skills of abstraction.
- Specialization
  - Allows to capture specific features of a set of objects.
  - Requires a depth of knowledge of the domain.

Polymorphism

- The ability of different types of objects to respond to the same message in different ways is called polymorphism.
- Polymorphism helps to :
  - Design extensible software; as new objects can be added to the design without rewriting existing procedures.



Containment

- One object may contain another as a part of its attribute
  - Document contains sentences which contain words.
  - Computer system has a hard disk, processor, RAM, mouse, monitor, etc.
- Containment need not be physical
  - E.g. Computer system has a warranty.

Containment Vs Inheritance

- Containment is used:
  - When the features of an existing class are wanted inside a new class, but not its interface.
    - Computer system has a hard disk.
    - Car has an engine, chassis, steering wheel.
- Inheritance is used:
  - When it is necessary that the new type has to be the same type as the base class.
    - Computer system is an electronic device.
    - Car is a vehicle.