C++ Classes

- The class is the cornerstone of C++
  - It makes possible encapsulation, data hiding and inheritance
- Type
  - Concrete representation of a concept
    - Eg. **float** with operations like -, *, + (math real numbers)
- Class
  - A user defined type
  - Consists of both data and methods
  - Defines properties and behavior of that type
- Advantages
  - Types matching program concepts
    - Game Program (Explosion type)
  - Concise program
  - Code analysis easy
  - Compiler can detect illegal uses of types
- Data Abstraction
  - Separate the implementation details from its essential properties

C++ Classes

- Information hiding
  - To prevent the internal representation from direct access from outside the class
- Access Specifiers
  - public
    - may be accessible from anywhere within a program
  - private
    - may be accessed only by the member functions, and friends of this class
  - protected
    - acts as public for derived classes
    - behaves as private for the rest of the program

C++ Classes

Header

```
class class_name
{
    permission_label:
        member;
    permission_label:
        member;
    ...
};
```

Body

```
class Rectangle
{
    private:
        int width;
        int length;
    public:
        void set(int w, int l);
        int area();
};
```

Classes and Objects

```
class Rectangle
{
   private:
      int width;
      int length;
   public:
      void set(int w, int l);
      int area();
};
```

Objects: Instance of a class

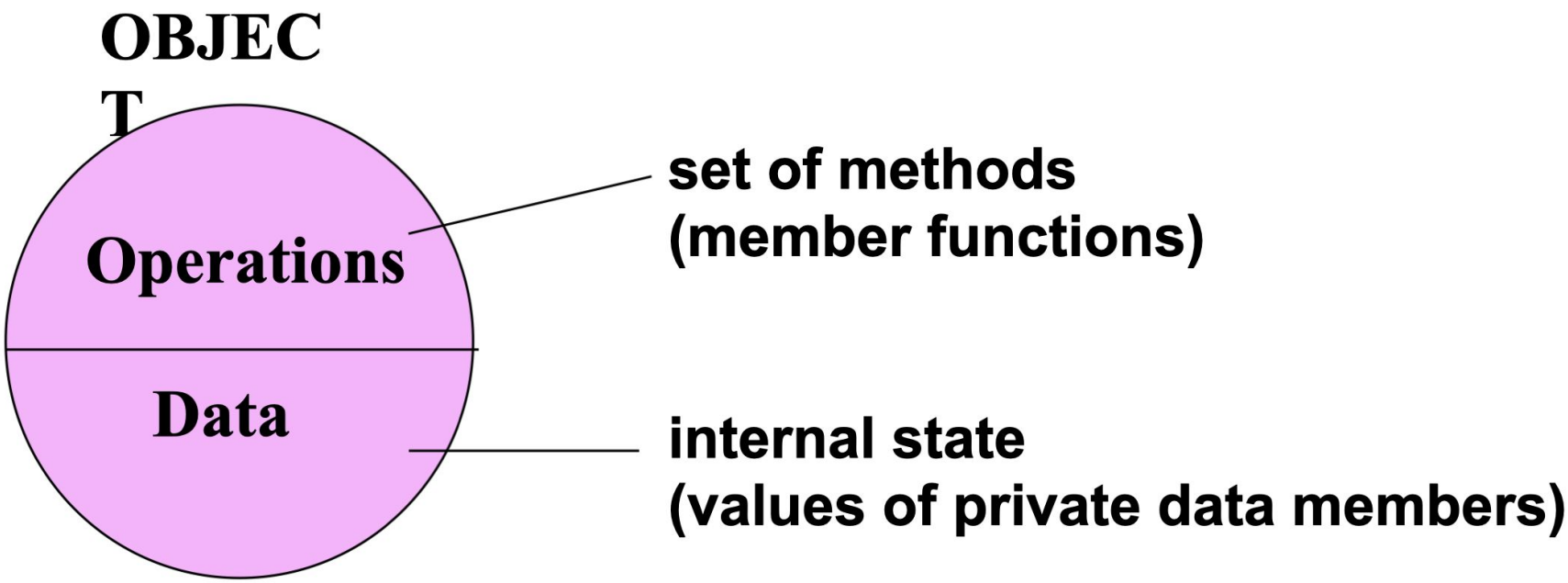
```
Rectangle  r1;
Rectangle  r2;
Rectangle  r3;

   ...
   ...
```

```
int   a;
```

Objects

What is an object?



Objects

```
class Rectangle
{
   private:
      int width;
      int length;
   public:
      void set(int w, int l);
      int area();
};
```

r1 is statically allocated

```
main()
{
   Rectangle r1;

   r1.set(5, 8);
}
```

r1

width = 5
length = 8

Data Members

- Can be of any type, built-in or user-defined
- *non-static* data member
  - Each class object has its own copy
- *static* data member
  - Acts as a global variable
  - One copy per class type, e.g. counter

Class Methods

- Used to
  - access the values of the data members (**accessor**)
  - perform operations on the data members (**implementor**)
- Are declared inside the class body
- Their definition can be placed inside the class body, or outside the class body
- Can access both public and private members of the class
- Can be referred to using dot or arrow member access operator

Const Methods

- **const** member function
  - declaration
    - *return_type func_name* (*para_list*) const;
  - definition
    - *return_type func_name* (*para_list*) const { … }
    - *return_type class_name* :: *func_name* (*para_list*) const { … }
  - Makes no modification about the data members (safe function)
  - It is illegal for a const member function to modify a class data member

Class declaration & Constructors

```
class  Time
{
 public :
   void     Set ( int  hours , int  minutes , int  seconds ) ;
   void       Increment ( ) ;
   void       Write ( )  const ;
   Time    ( int  initHrs, int  initMins,  int  initSecs ) ;   // constructor
   Time    ( ) ;                                  // default constructor
 private :
   int            hrs ;
   int            mins ;
   int            secs ;
} ;
```


Class declaration, implementation & usage

```
#include <iostream.h>

class circle
{
  private:
    double radius;

  public:
    void store(double);
    double area(void);
    void display(void);

};
```

```
// member function definitions

void circle::store(double r)
{
  radius = r;
}
double circle::area(void)
{
  return 3.14*radius*radius;
}
void circle::display(void)
{
  cout << "r = " << radius << endl;
}
```

```
int main(void) {
  circle c;   // an object of circle class
  c.store(5.0);
  cout << "The area of circle c is " << c.area() << endl;
  c.display();
}
```