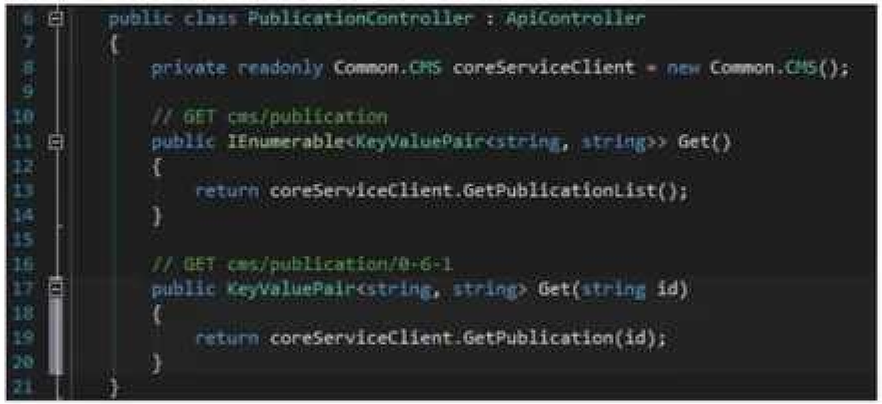
WHO WOULD WIN?

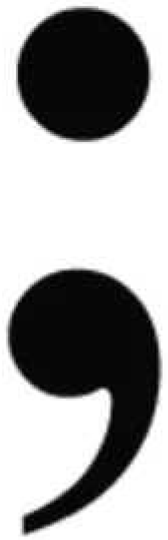


```
public class PublicationController : ApiController
{
    private readonly Common.CMS coreServiceClient = new Common.CMS();

    // GET cms/publication
    public IEnumerable<KeyValuePair<string, string>> Get()
    {
        return coreServiceClient.GetPublicationList();
    }

    // GET cms/publication/0-6-1
    public KeyValuePair<string, string> Get(string id)
    {
        return coreServiceClient.GetPublication(id);
    }
}
```

A large complex script of code written by Professional computer programmers And designers



One no-mouth winky boi

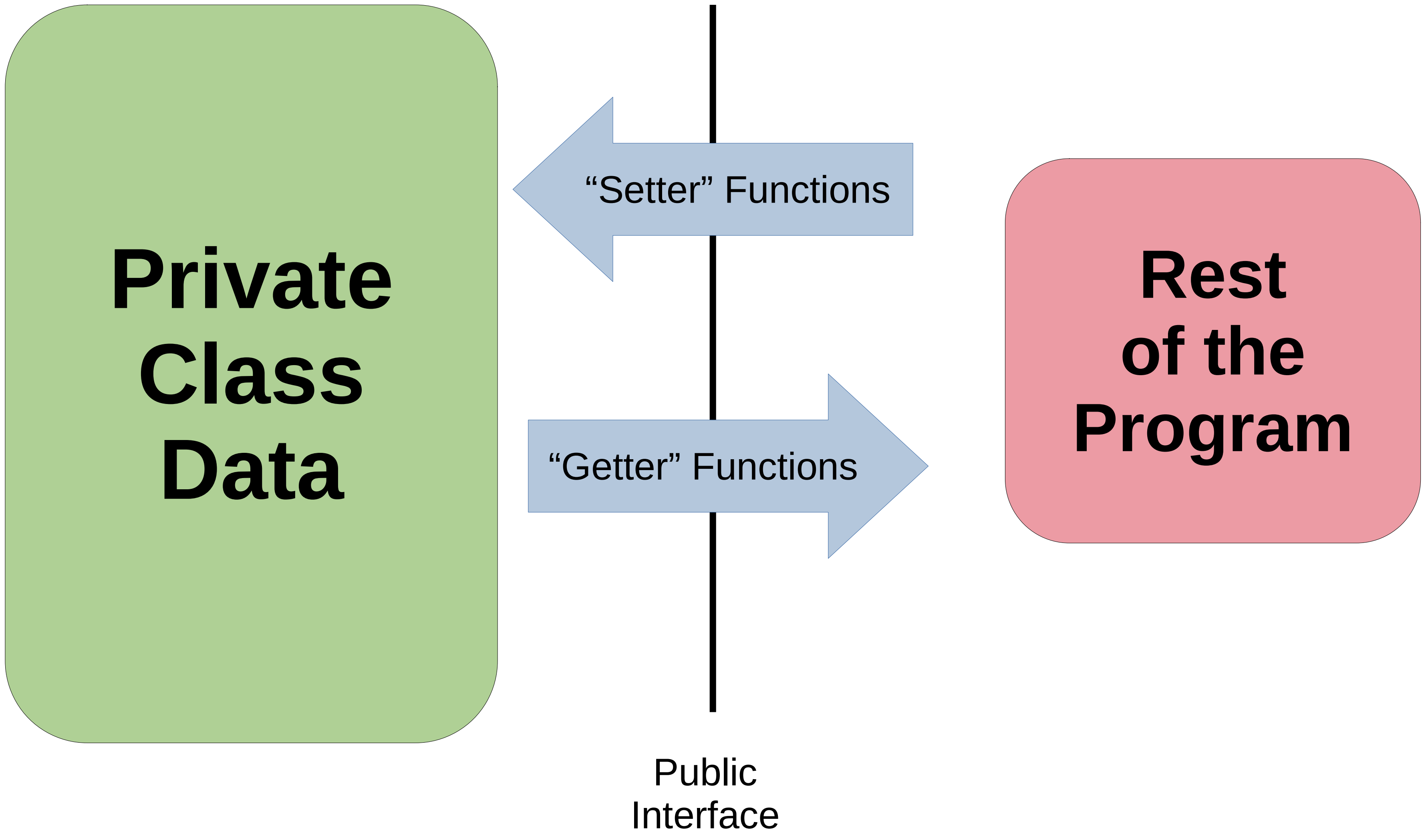
Defining Classes

CS2308
Gentry Atkinson

Structs vs. Classes

- **structs** and **classes** create new data types by combining primitive data types.
- Users have access to all members of a **struct.**
- We can write separate functions to manipulate the members of a **struct.**
- **Classes** let us combine data and functions into a single package.

What is a Class?



What is a Class?

- A **class** defines an abstraction used in a program not just in terms of what they “know”, but also what they “do”.
- Restricts access to private data to ensure that it is used “safely” and “correctly”.
- The **class** is the definition. An instance of a class in memory is called an **object.**

Limited Access

- The keywords **private** and **public** can be used define which class members can be accessed from outside the **class**.
- **private** members can be accessed by the **class** itself, but not from outside the class.
- Data should be kept **private** as much as possible.
- **public** funcitons are used to access **private** data.

Example 1

```
class Circle{
    private:
        const float PI = 3.14159;
    public:
        float radius, circum, area;
};

int main(int argc, char** argv){
    Circle c;
    cout << c.PI << endl;
} //causes an error
```

Member Functions

- **classes** collect both data and functions into one package.
- Member functions can be called **methods.**
- Member functions are called using the member access dot . operator, just like member data.
- Member functions can be public or private.
- Member functions have access to private class members.

Example 2

```
class Circle{
    private:
        const float PI = 3.14159;
    public:
        float radius, circum, area;
        void setMembers(float radius){
            this->radius = radius;
            area = PI*(radius*radius);
            circum = PI*(2*radius);
        }
};
```

Example 2 cont.

```
int main(int argc, char** argv){
    Circle c;
    c.setMembers(5);
    cout << c.area << endl;
} //try to guess the output
```

Getters and Setters

- Using member functions to initialize, update, and retrieve the values stored in member variables means that we can always insure that the values are “correct”.
- **Getters**: retrieve a private value from a class. Also called **accessors.**
- **Setters**: alter the value of a private member of a class. Also called **mutators.**
- Data and functions should always be as private as possible.

Example 3

```
class Circle{
    private:
        const float PI = 3.14159;
        float radius, circum, area;
    public:
        void **setRadius**(float r){radius = r;}
       void **setArea**(){ area =
PI*(radius*radius);}
        void **setCircum**(){circum =
PI*(2*radius);}


        float **getArea**(){return area;}
};
```

```
int main(int argc, char** argv){
    Circle c;
    c.setRadius(5);
    c.setArea();
    cout << c.getArea() << endl;
} //try to guess the output
```

Constructors and Destructors

- Constructors run automatically when a class is instantiated.
- Destructors run automatically when a class is deleted.
- Constructors must have the same name as the class.
- Destructors are named ~ + Class Name.
- As member functions, constructors and destructors can access private members, but must be public.

Example 4

```
class Circle{
    private:
        const float PI = 3.14159;
        float radius, circum, area;
    public:
        Circle(float r){
            radius = r;
            area = PI*(radius*radius);
            circum = PI*(2*radius);
        }
        float getArea(){return area;}
};
```

```
int main(int argc, char** argv){
    Circle c(5);
    cout << c.getArea() << endl;
} //try to guess the output
```

Class vs. Object

- A **class** is a definition of a new data type, like a **struct**.
- Create a variable with the type of a class is called “**instantiating**” a class.
- An **object** is a variable with the type of a class which exists in memory.

Questions or Comments?