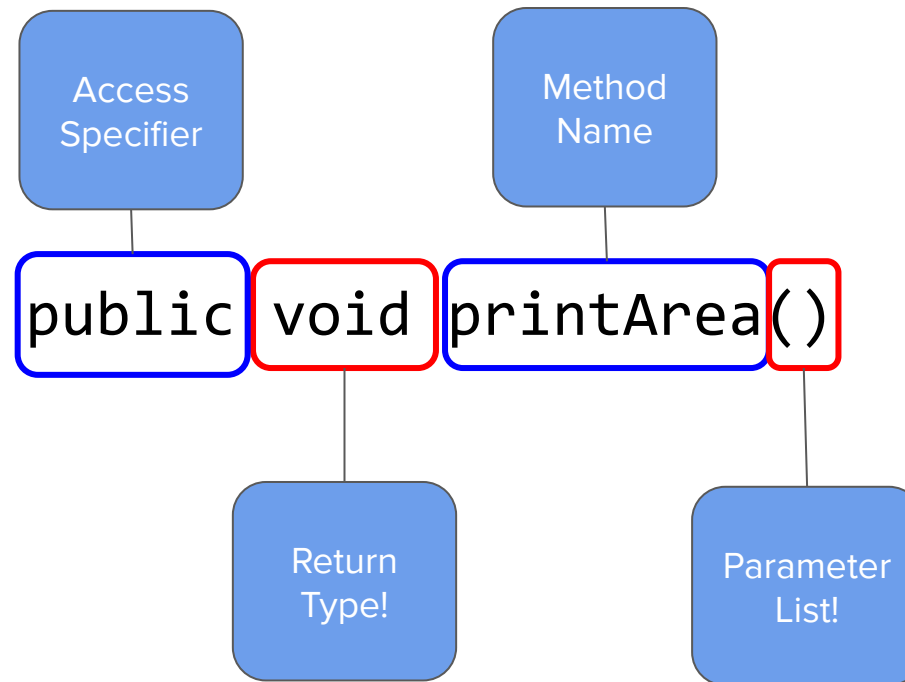


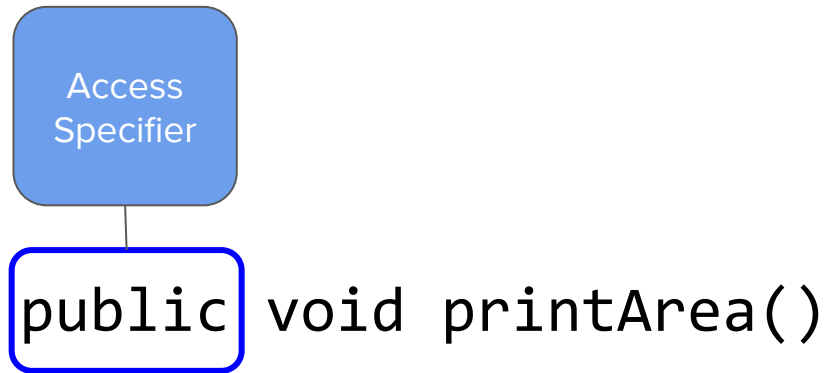
AP Computer Science A:

Anatomy of Classes

Recap: Writing Method Declarations



Recap: Writing Method Declarations



Access Specifiers

Access Specifiers determine whether classes, data, constructors and methods can be accessed **outside** of the declaring class

Access Specifiers

Types of Access:

public: allows access from classes outside the declaring class.

private: restricts access to the declaring class.

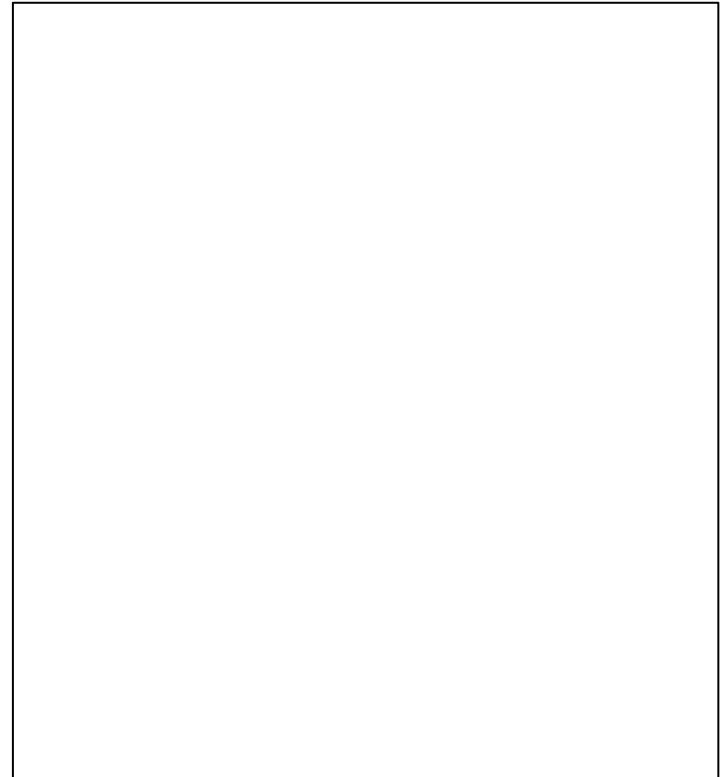
Declaring Class

Rectangle.java Declaring Class

```
public class Rectangle
{

}
}
```

MyProgram.java



Declaring Class

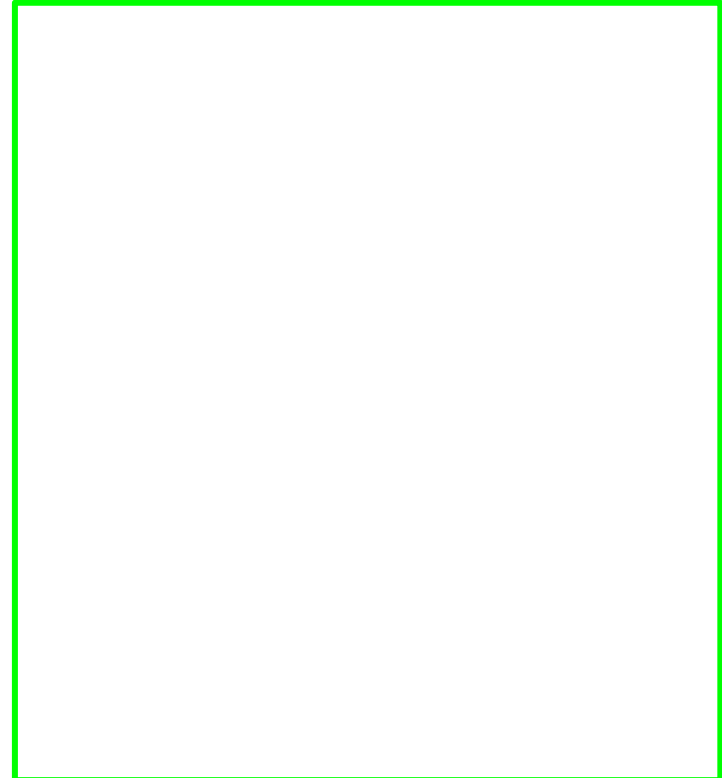
Rectangle.java Declaring Class

```
public class Rectangle
{

    /*public methods, variables
    and constructors written in
    Rectangle can be used in
    MyProgram.java and in
    Rectangle.java*/

}
```

MyProgram.java



Declaring Class

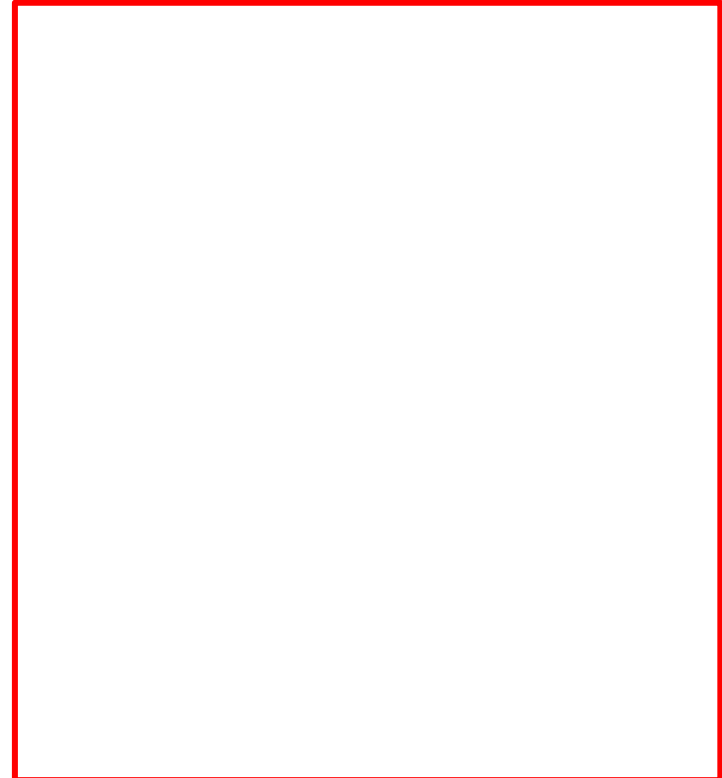
Rectangle.java Declaring Class

```
public class Rectangle
{

    /*private methods,
    variables and constructors
    written in Rectangle cannot
    be used in MyProgram.java,
    but can be used and
    accessed in
    Rectangle.java*/

}
```

MyProgram.java



public Access

Classes and constructors are designated as **public** so that they can be accessed outside of the class file!

```
public class Rectangle
{
    public Rectangle()
    {
```

private vs. public

```
public Rectangle()  
{  
    //implementation  
}
```

MyProgram.java

```
Rectangle rect = new Rectangle();
```

```
private Rectangle()  
{  
    //implementation  
}
```

MyProgram.java

```
Rectangle rect = new Rectangle();
```

Errors:

```
MyProgram.java: Line 6: Rectangle() has private access in  
Rectangle
```

private access

We use `private` access for methods and data in order to manage complexity.

We do so by controlling how users can interact with objects and hiding implementation details that are unnecessary for users.

Encapsulation

This process of hiding implementation details is referred to as **encapsulation**.

Encapsulation vs Abstraction

Encapsulation is different than Abstraction because it is concerned with hiding the internal **state** of an object, whereas Abstraction is concerned with hiding the details of how something works.

Instance Variables

Access to attributes should be kept internal to the class. Instance variables are designated as **private** so as to prevent users from directly manipulating the state of an object.

```
public class Rectangle
{
    private int width;
    private int height;
}
```

Instance Variables

```
public class Rectangle
{
    int width;
    int height;
}
```

```
Rectangle rect = new Rectangle(10,5);
System.out.println(rect.width);
rect.width = 16;
System.out.println(rect.width)
```

```
10
16
```

```
public class Rectangle
{
    private int width;
    private int height;
}
```

```
Rectangle rect = new Rectangle(10,5);
System.out.println(rect.width);
rect.width = 16;
System.out.println(rect.width)
```

Errors:

MyProgram.java: Line 7: width has private access in Rectangle

MyProgram.java: Line 8: width has private access in Rectangle

MyProgram.java: Line 9: width has private access in Rectangle

Instance Variables

```
public class Rectangle
{
    int width;
    int height;
}
```

```
Rectangle rect = new Rectangle(10,5);
System.out.println(rect.width);
rect.width = 16;
System.out.println(rect.width)
```

We can access public instance variables by using the object.variable format

```
10
16
```


Instance Variables

We don't want users changing values we want to keep constant or well organized!

MyProgram.java

```
public class Student
{
    int eyes = 2;
    String hairColor;
    String eyeColor;

    public Student(String hair, String eyeCol)
    {
        if(hair.equals("brown") || hair.equals("blonde"))
        {
            hairColor = hair;
        }
        eyeColor = eyeCol;
    }
}
```

```
Student person = new Student("brown","blue");
person.eyes = 100;
person.hairColor = "cat";
```

Instance Variables

We don't want users changing values we want to keep constant or well organized!

MyProgram.java

```
public class Student
{
    int eyes = 2;
    String hairColor;
    String eyeColor;

    public Student(String hair, String eyeCol)
    {
        if(hair.equals("brown") || hair.equals("blonde"))
        {
            hairColor = hair;
        }
        eyeColor = eyeCol;
    }
}
```

```
Student person = new Student("brown","blue");
person.eyes = 100;
person.hairColor = "cat";
```

hairColor should only be brown or blonde, and eyes should always be 2. public access allows users to change that!

Instance Variables

We don't want users changing values we want to keep constant or well organized!

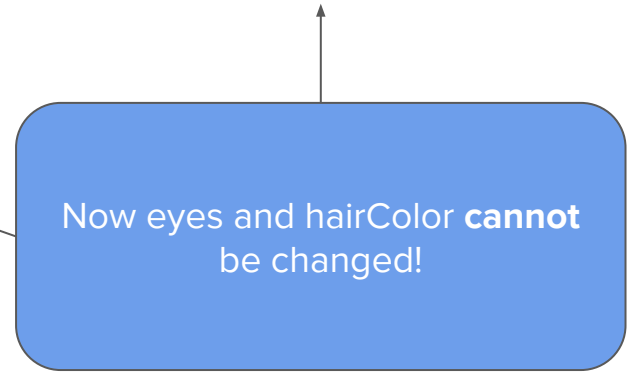
```
public class Student
{
    private int eyes = 2;
    private String hairColor;
    private String eyeColor;

    public Student(String hair, String eyeCol)
    {
        if(hair.equals("brown") || hair.equals("blonde"))
        {
            hairColor = hair;
        }
        eyeColor = eyeCol;
    }
}
```

MyProgram.java

```
Student person = new Student("brown","blue");
person.eyes = 100;
person.hairColor = "cat";
```

Now eyes and hairColor **cannot** be changed!



Accessors and Mutators

Rather than allow users to manipulate data directly, we use **accessor** (getter) and **mutator** (setter) methods to control what aspects of an object can be altered:

```
public int getWidth()  
{  
    return width;  
}
```

```
public void setWidth(int newWidth)  
{  
    width = newWidth;  
}
```

Data Access

Programmers have to choose whether data should be **accessible, modifiable, both, or neither.**

Data Access

Programmers have to choose whether data should be **accessible**, **modifiable**, **both**, or **neither**.

```
private int width;
```

```
public int getWidth()
```

```
{
```

```
    return width;
```

```
}
```

Data Access

Programmers have to choose whether data should be **accessible**, **modifiable**, **both**, or **neither**.

```
private int width;
```

```
public void setWidth(int newWidth)
{
    width = newWidth;
}
```

```
public int width;
```

Data Access

Programmers have to choose whether data should be **accessible, modifiable, both, or neither.**

```
private int width;

public void setWidth(int newWidth)
{
    width = newWidth;
}
```

```
private int width;

public int getWidth()
{
    return width;
}
```


Data Access

Programmers have to choose whether data should be **accessible, modifiable, both, or neither**.

```
private int width;
```

Savings Account Class

```
public class SavingsAccount
{
    private Date theDate;
    private int accountTotal;
    private Date interestDate;

    public SavingsAccount(int deposit)
    {
        accountTotal = deposit;
        theDate = new Date();
    }

    public int getBalance()
    {
        return accountTotal;
    }

    public Date getDate()
    {
        return theDate;
    }

    private void checkDate()
    {
        interestDate = new Date();
    }
}
```

DATE	DESCRIPTION	WITHDRAWALS	DEPOSITS	BALANCE
03-10-16	ATMW	**21.25		**474.11
03-10-16	ATMF	**1.50		**472.61
03-10-20	DEBP	**2.99		**469.62
03-10-21	WEBP	**300.00		**169.62
03-10-22	ATMW	**100.00		**69.62
03-10-23	DEBP	**29.08		**40.54
03-10-24	DEBR		**2.99	**43.53
03-10-27	TELP	**6.77		**36.76
03-10-28	PYRL		**694.81	**731.57
03-10-30	WEBT		**50.00	**781.57
Please refer to the back cover for the list of common transaction codes.			Please verify your account activity regularly. If there is an error, notify the bank within 45 days.	

Savings Account

What are some methods we might want users to have access to?

Savings Account

What are some methods we might want users to have access to?

Account Information:

```
public int getAccountBalance()
```

Removing Money from Account:

```
public void withdrawal(int money)
```

Adding Money to Account:

```
public void deposit(int money)
```

Get Transaction History:

```
public String getTransactionHistory(int month)
```

Savings Account

What are some methods users might NOT need to access?

Savings Account

What are some methods users might NOT need to access?

Adding Interest to Money in Account:

```
private int addInterest()
```

How Long the Account has been Opened:

```
private int daysOpen(Date theDate)
```

Savings Account

What are some methods users might NOT need to access?

Adding Interest to Money in Account:

```
private int addInterest()
```

How Long the Account has been Opened:

```
private int daysOpen(Date theDate)
```

Setting methods to **private** does not mean that we don't trust users, it's simply that they don't **NEED** access to those methods in order to implement the class correctly.

Savings Account

private methods can only be used within the class they are declared!

Adding Interest to Money in Account:

```
private int addInterest()
```

How Long the Account has been Opened:

```
private int daysOpen(Date theDate)
```

```
public class SavingsAccount  
{
```

```
    private void checkInterest(Date theDate)  
    {  
        if(daysOpen(theDate) > 30)  
        {  
            int interest = addInterest();  
            deposit(interest);  
        }  
    }  
}
```


Now It's Your Turn!

Concepts Learned this Lesson

Term	Definition
public	Allows access to data and methods from classes outside the declaring class.
private	Restricts access to data and methods to the declaring class.
Encapsulation	The process of hiding the implementation details of a class from the user
Accessor Methods	Methods used to access instance variable and object data. Also referred to as getter methods.
Mutator Methods	Methods used to change or manipulate instance variable or object data. Also referred to as setter methods.

Standards Covered

- (LO) MOD-2.A Designate access and visibility constraints to classes, data, constructors, and methods.
- (EK) MOD-2.A.1 The keywords public and private affect the access of classes, data, constructors, and methods.
- (EK) MOD-2.A.2 The keyword private restricts access to the declaring class, while the keyword public allows access from classes outside the declaring class.
- (EK) MOD-2.A.3 Classes are designated public.
- (EK) MOD-2.A.4 Access to attributes should be kept internal to the class. Therefore, instance variables are designated as private.
- (EK) MOD-2.A.5 Constructors are designated public.
- (EK) MOD-2.A.6 Access to behaviors can be internal or external to the class. Therefore, methods can be designated as either public or private.
- (LO) MOD-3.A Designate private visibility of instance variables to encapsulate the attributes of an object.
- (EK) MOD-3.A.1 Data encapsulation is a technique in which the implementation details of a class are kept hidden from the user.
- (EK) MOD-3.A.2 When designing a class, programmers make decisions about what data to make accessible and modifiable from an external class. Data can be either accessible or modifiable, or it can be both or neither.
- (EK) MOD-3.A.3 Instance variables are encapsulated by using the private access modifier.
- (EK) MOD-3.A.4 The provided accessor and mutator methods in a class allow client code to use and modify data.