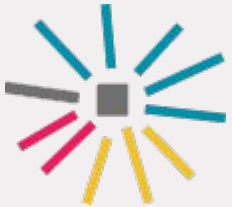
METHODS

PALIMPSEST



METHODS

What are the verbs of a program?

- Definition
- EXERCISE
- Declaration
- EXERCISE
- Naming
- Return type
- Scope

METHODS

METHOD DEFINITION

An code object that has behaviors and data.

```
// Class to express myself to the world.

public class YoWorldIExist
{
    // Method to print string to console.
    public void Speak( string phrase )
    {
        Debug.Log( phrase );
    }
}
```

Hey there, girl. Let’s learn about methods!

Hey there, girl. Let's learn about methods!

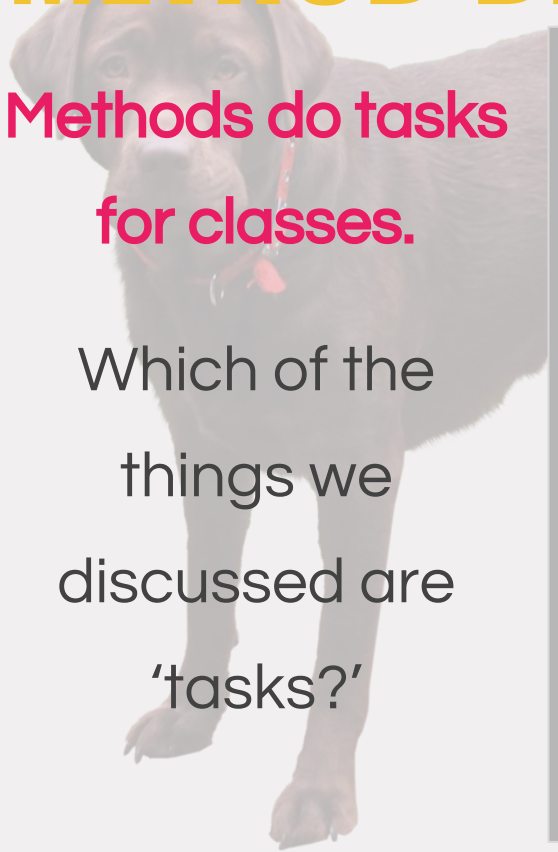
DOG CLASS

```
// Class to define a dog.

public class Dog
{
    //Variables describing a dog.
    private int ageInYears;

    // Method making the dog speak.
    public void Speak( string phrase )
    {
        Debug.Log( phrase );
    }
}
```

WHAT DATA DESCRIBES THIS DOG?

WHAT DOES THIS DOG DO?

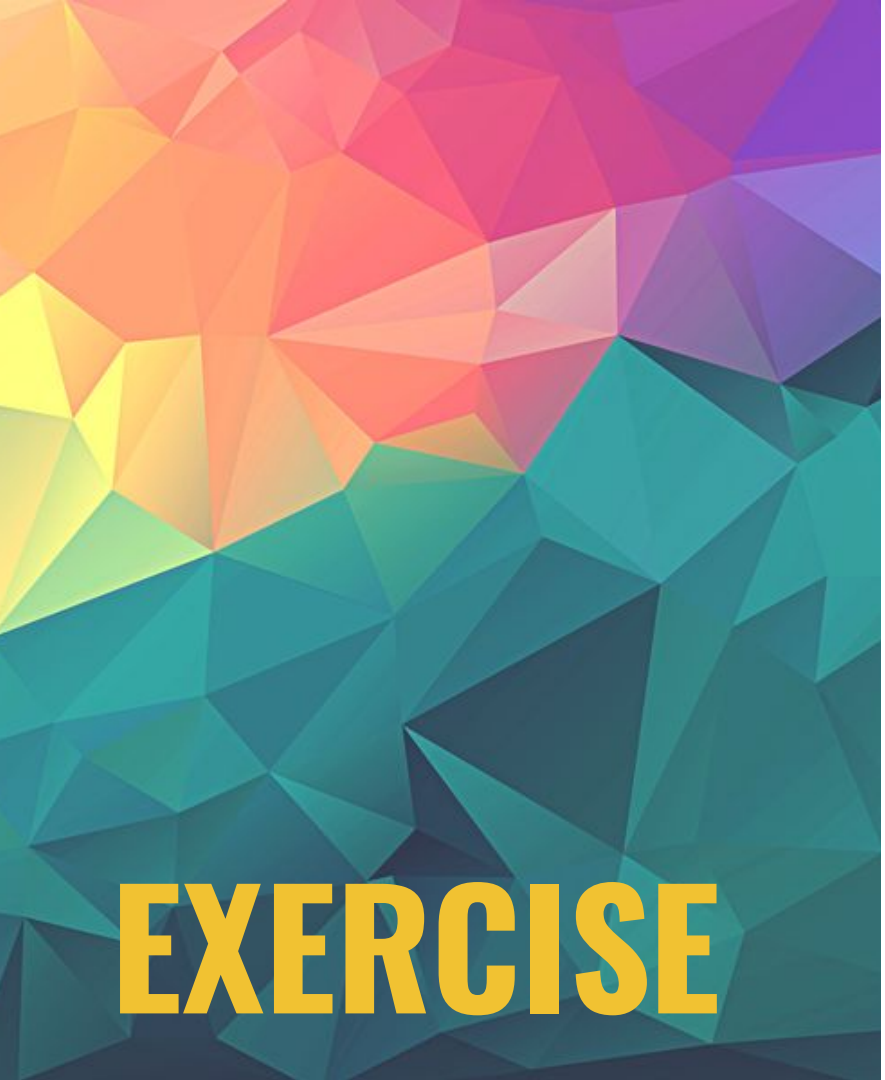
METHOD DEFINITION



Methods do tasks for classes.

Which of the things we discussed are ‘tasks?’

```
// Class to define a dog.

public class Dog
{
    //Variables describing a dog.
    private int ageInYears;

    // Method making the dog speak.
    public void Speak( string phrase )
    {
        Debug.Log( phrase );
    }
}
```



- Think of a character
- Think of behaviors
- Using pseudocode, make those behaviors "methods"
- Bonus - what are some properties of this character? How might you express those?

METHOD BENEFITS

- Helps manage complexity
- Makes it easier to build large programs
- Makes it easier to debug programs

METHOD BENEFITS

- Methods are self-contained

- Meaning: You can use a method someone else wrote, without knowing how it does what it does.

METHOD DECLARATION

1. Modifiers
2. Return Type
3. Name
4. Parameter type
5. Parameter name
6. Method body
7. Method variable declarations
8. Method statements
9. Return statement
10. Return value

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

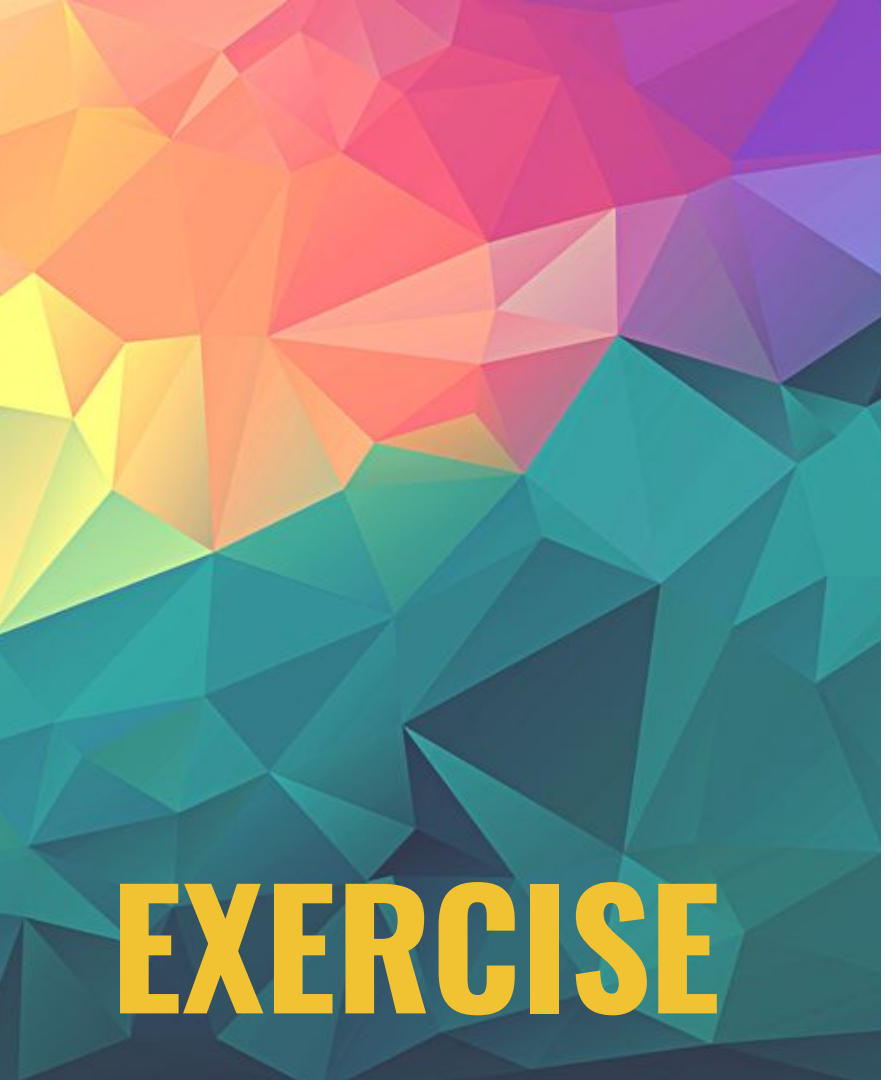
METHOD DECLARATION

1. Modifiers
2. Return Type
3. Name
4. Parameter type
5. Parameter name
6. Method body
7. Method variable declarations
8. Method statements
9. Return statement
10. Return value

```
// Method to guard home from skeery cars.
    1     2     3     4     5
   public int Guard( int numCars )
  {
     7 int numAnnoyedNeighbors = 0;

     8 for (i=0; i < numCars; i++)
       {
         8 Debug.Log("Bark!");
         8 annoyedNeighbor++;
       }
     9 return numAnnoyedNeighbors;
  }
```

6

10



- Use Unity and C# to write a method that returns a random number.
- Display the return value in the console.

GOOD METHOD HYGIENE

NAMING IS IMPORTANT

- Same rules as naming variables
- Name clearly describes the method's task
  - helps debugging
  - helps others read your code
- EXAMPLE:

  public static double calculateBonus(int level)

- Parentheses indicate a method
  - VARIABLE: speedingCar
  - METHOD: speedingCar()

GOOD METHOD HYGIENE

PROGRAMMING PRACTICES

Methods should do ONE thing.

Good names help!

```
calculateTaxAndPrintReturnAndSaveFile(float rate);
```

This method is probably doing too much

```
calculateTax(float rate);
```

This method is doing fine.

GOOD METHOD HYGIENE

PROGRAMMING PRACTICES

Avoid duplication

Avoid duplication

Avoid duplication

Avoid duplication

Avoid duplication

Avoid duplication

Avoid duplication

Avoid duplication

Avoid duplication

GOOD METHOD HYGIENE

PROGRAMMING PRACTICES

Try not to repeat code.

Anything that repeats, might need to be made into a method.

WHY?

RETURNING VALUES

ONLY ONE return value

OR none: void

- void: means nothing
- A method that returns void returns...

RETURNING VALUES

ONLY ONE return value

OR none: void

- void: means nothing
- A method that returns void returns...

PARAMETERS

```
// Method to spawn colored spheres.

public void SpawnSpheres(int numSpheres, Vector3
                         location, color sphereColor )
{

    // fill body in later

}
```

PARAMETERS

```
// Method to spawn colored spheres.

public void SpawnSpheres(int numSpheres, Vector3
                         location, Color sphereColor )
{

    // fill body in later

}
```

- Any number of parameters
- Separate parameters with commas

PARAMETERS

```
// Method to spawn colored spheres.

public void SpawnSpheres(int numSpheres, Vector3
                         location, color sphereColor )
{

    // fill body in later

}
```

OR

NO PARAMETERS

```
// Method to roll a die.
public int rollDie()


//Method to grow all plants on screen

public void growPlants()
```

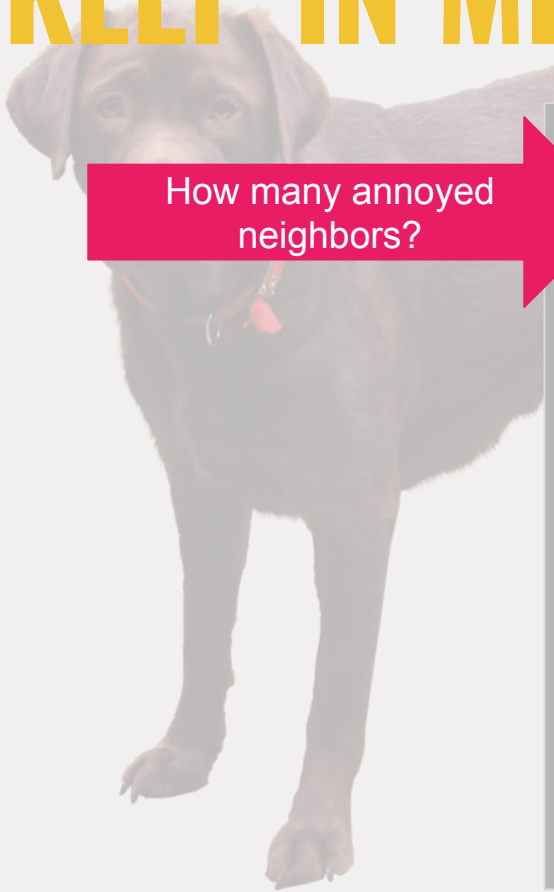
CALLING A METHOD


```
private int dieResult;

// roll die and assign result
dieResult = rollDie()


// it just rained, now grow plants

growPlants()
```

KEEP IN MIND

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

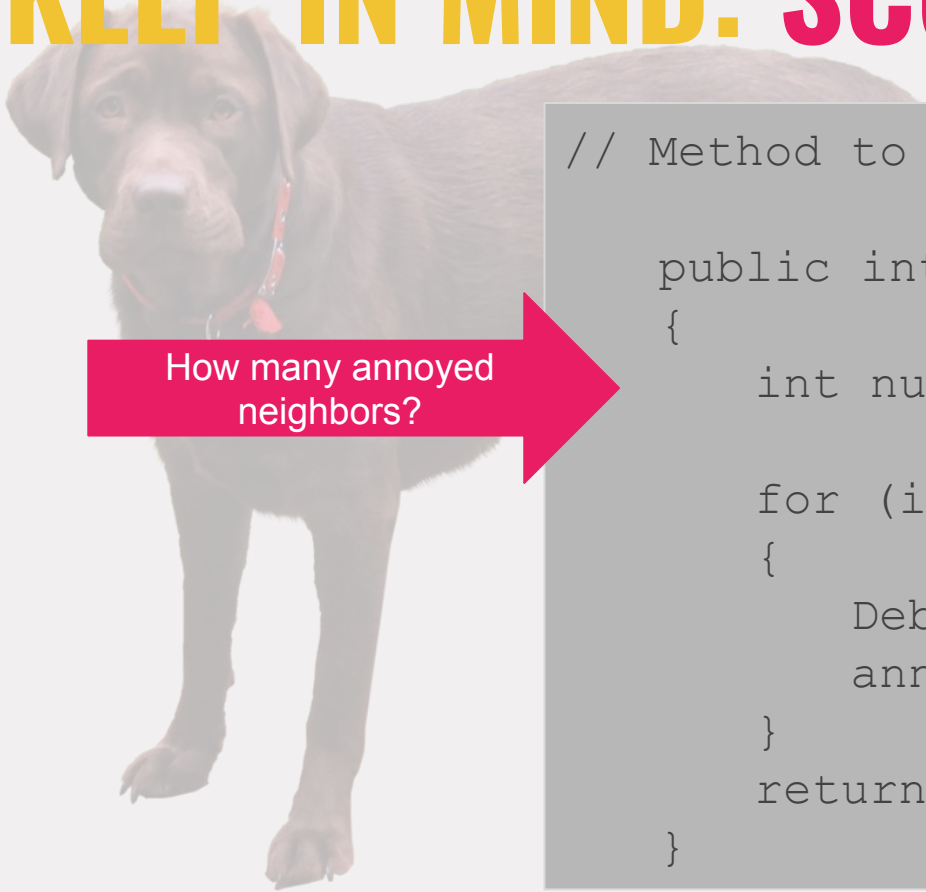
KEEP IN MIND: SCOPE



How many annoyed neighbors?

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

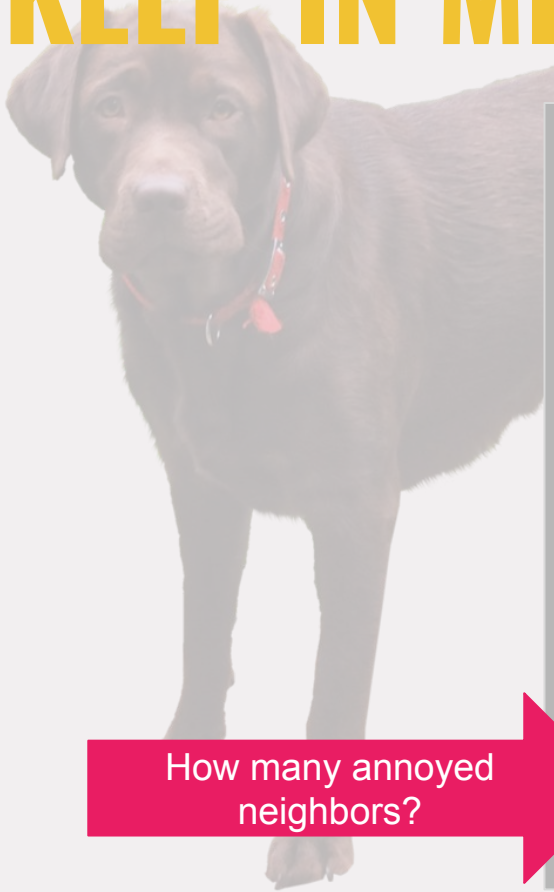
KEEP IN MIND: SCOPE



How many annoyed neighbors?

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

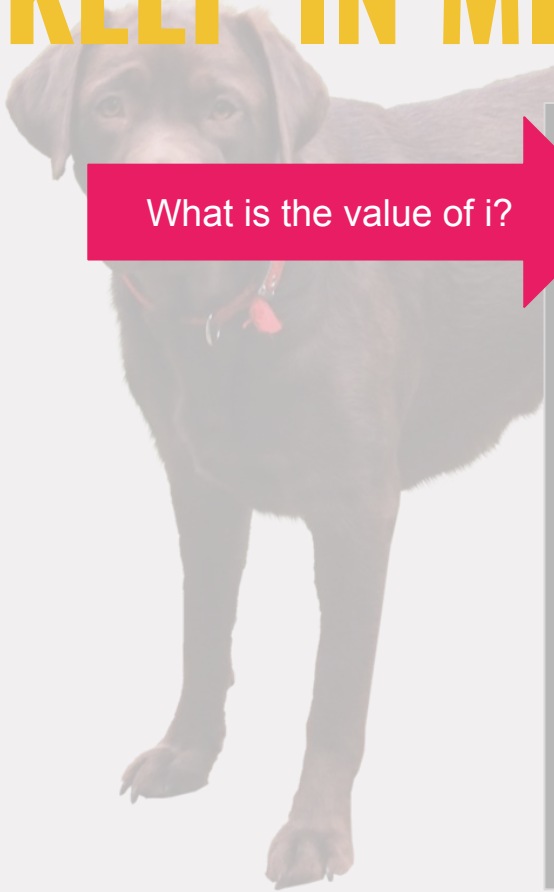
KEEP IN MIND: SCOPE



How many annoyed neighbors?

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

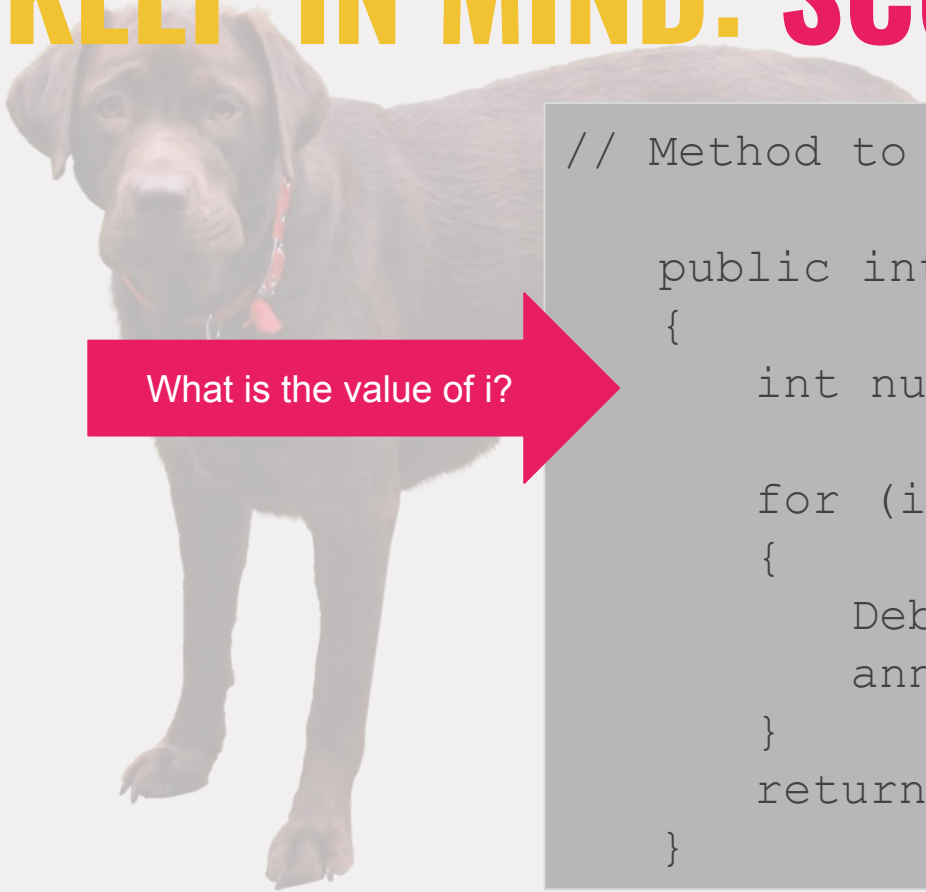
KEEP IN MIND: SCOPE



What is the value of i?

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

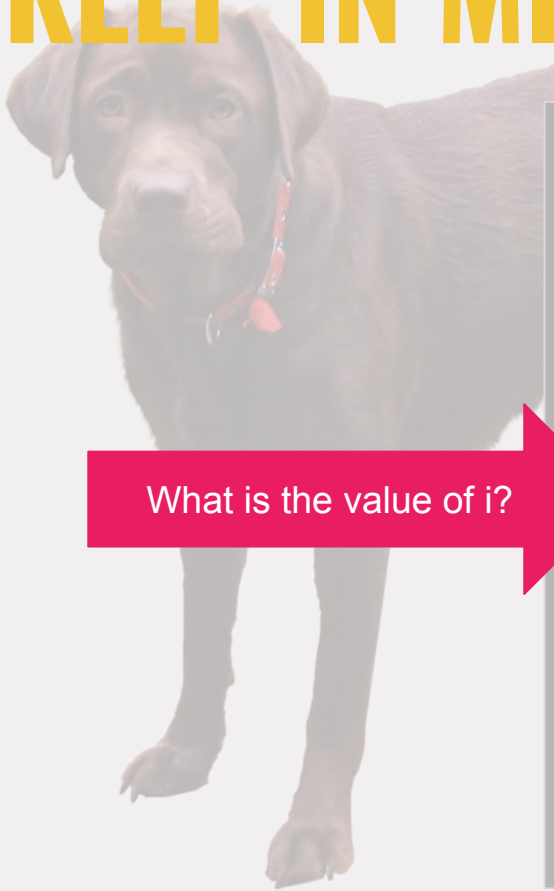
KEEP IN MIND: SCOPE



What is the value of i?

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

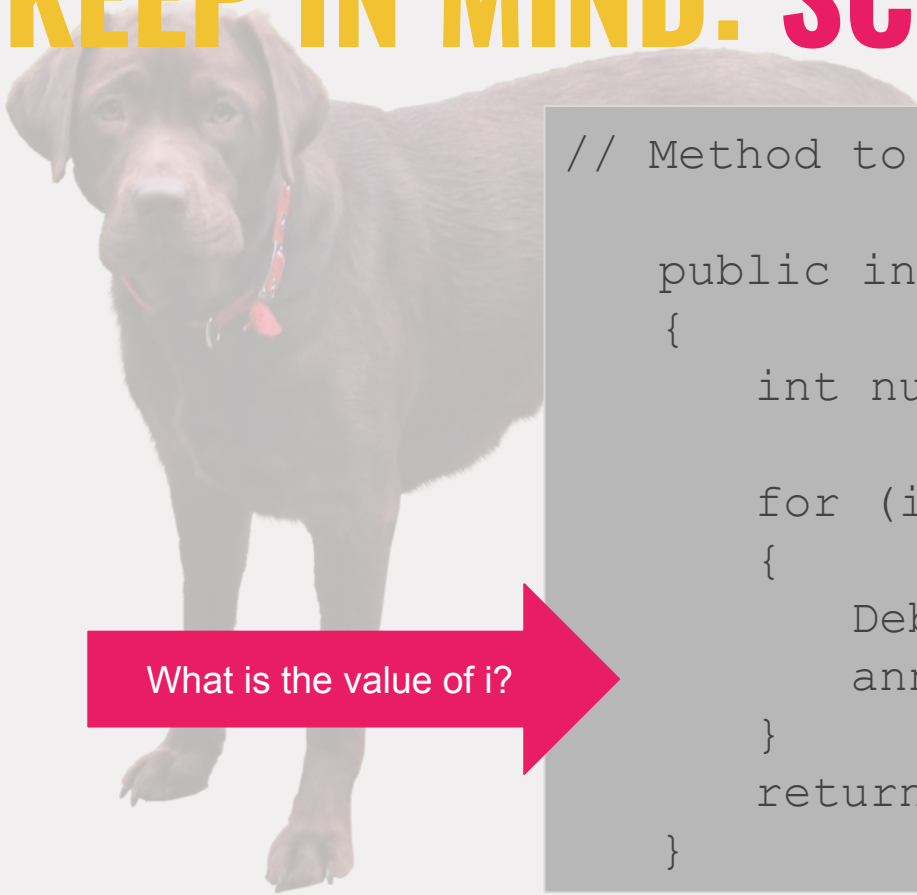
KEEP IN MIND: SCOPE



What is the value of i?

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

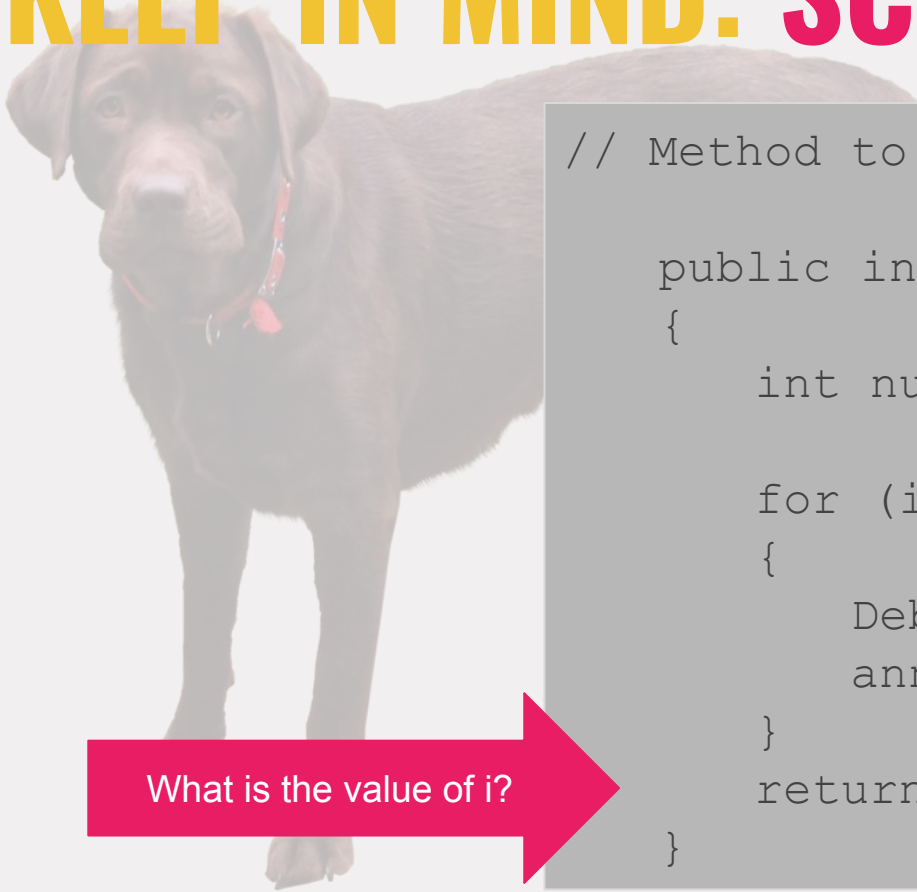
KEEP IN MIND: SCOPE



What is the value of i?

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

KEEP IN MIND: SCOPE



What is the value of i?

```
// Method to guard home from skeery cars.

    public int Guard( int numCars )
    {
        int numAnnoyedNeighbors = 0;

        for (i=0; i < numCars; i++)
        {
            Debug.Log("Bark!");
            annoyedNeighbor++;
        }
        return numAnnoyedNeighbors;
    }
```

SCOPE

DEFINITION AND USE

- Where a variable is declared determines its **scope**
- Scope is where that variable can referenced
  - reference: used, assigned, altered, read
- A variable declared inside a method cannot be used outside it.

METHODS

REVIEW

- Are code objects with data and behaviors
- Do tasks for classes
- They can take parameters
- They can return a single value
  - But don't have to: void
- They can be distinguished visually from variables by parentheses
- Make code easier to read
- Reduce duplication

METHODS

REVIEW

- Are code objects with data and behaviors
- Do tasks for classes
- They can take parameters
- They can return a single value
  - But don't have to: void
- They can be distinguished visually from variables by parentheses
- Make code easier to read
- Reduce duplication

METHODS

REVIEW

- Are code objects with data and behaviors
- Do tasks for classes
- They can take parameters
- They can return a single value
  - But don't have to: void
- They can be distinguished visually from variables by parentheses
- Make code easier to read
- Reduce duplication

METHODS

REVIEW

- Are code objects with data and behaviors
- Do tasks for classes
- They can take parameters
- They can return a single value
  - But don't have to: void
- They can be distinguished visually from variables by parentheses
- Make code easier to read
- Reduce duplication

METHODS

REVIEW

- Are code objects with data and behaviors
- Do tasks for classes
- They can take parameters
- They can return a single value
  - But don't have to: void
- They can be distinguished visually from variables by parentheses
- Make code easier to read
- Reduce duplication

METHODS

REVIEW

- Are code objects with data and behaviors
- Do tasks for classes
- They can take parameters
- They can return a single value
  - But don't have to: void
- They can be distinguished visually from variables by parentheses
- Make code easier to read
- Reduce duplication

METHODS

REVIEW

- Are code objects with data and behaviors
- Do tasks for classes
- They can take parameters
- They can return a single value
  - But don't have to: void
- They can be distinguished visually from variables by parentheses
- Make code easier to read
- Reduce duplication

NEXT UP

LAB