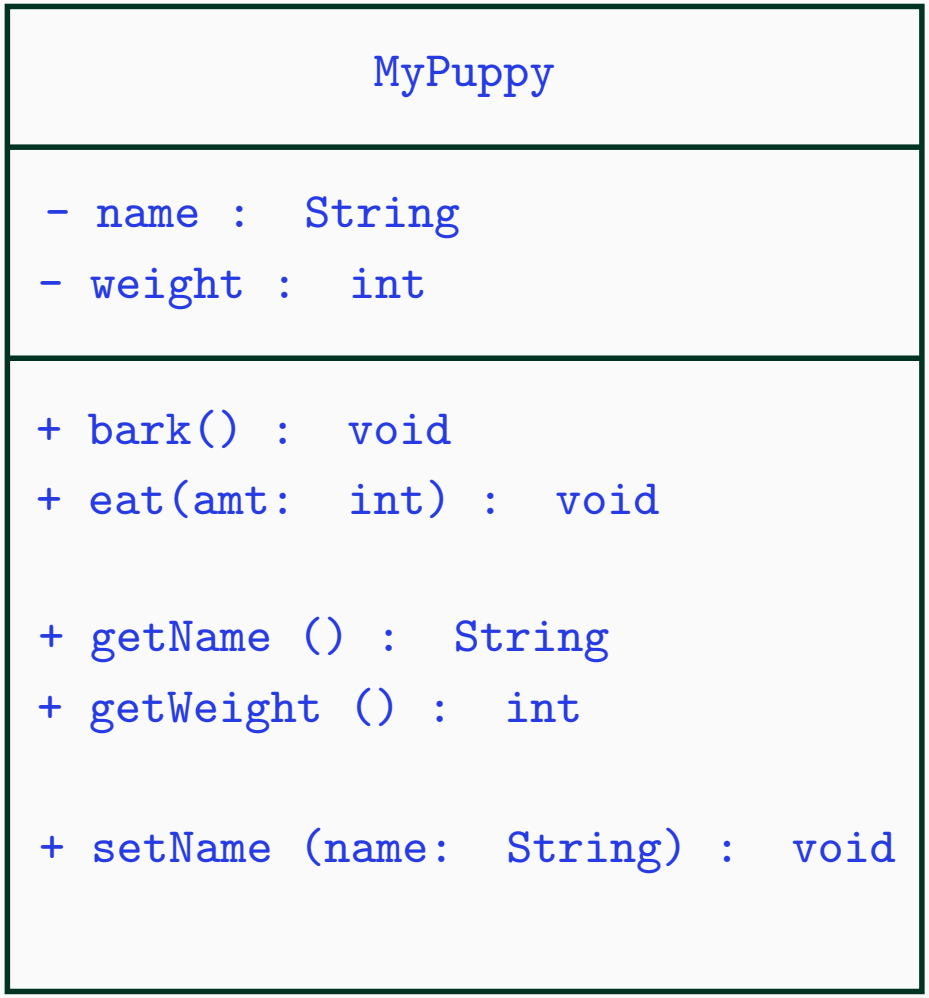
From UML to Coding

Shirley B. Chu

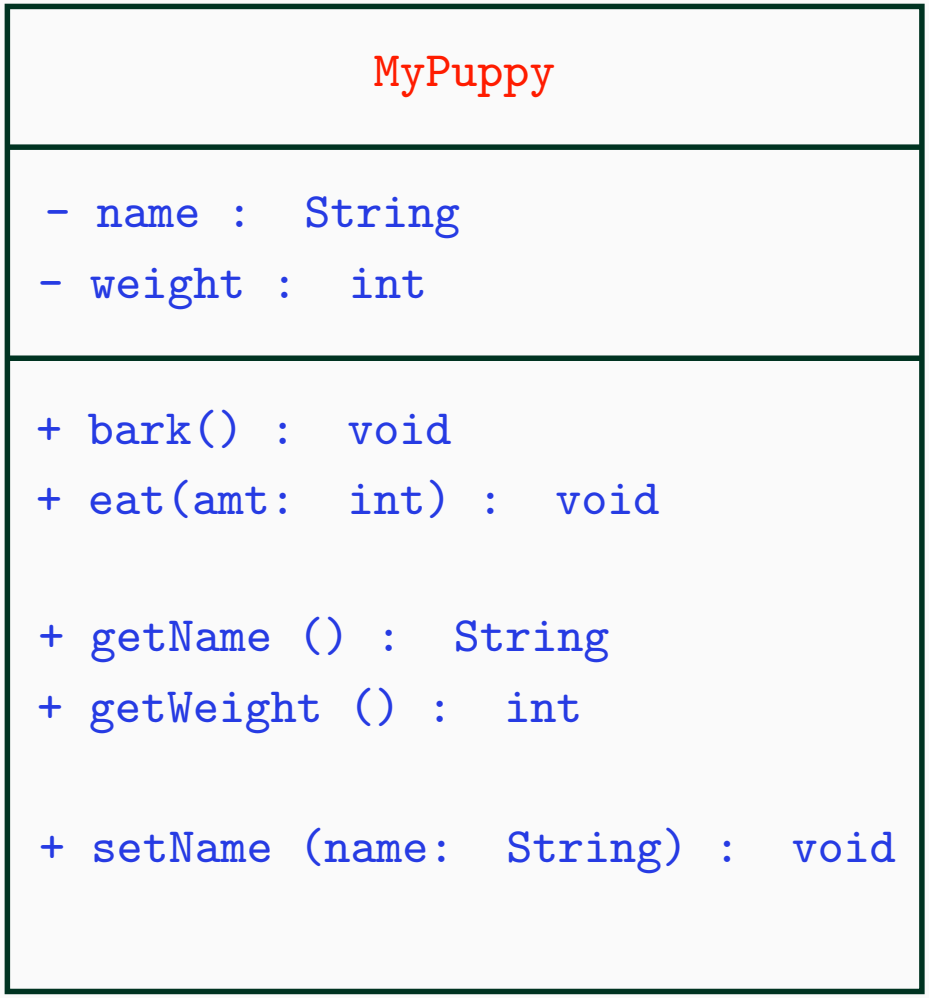
June 17, 2020

De La Salle University
College of Computer Studies

MyPuppy class (no explicit constructor)

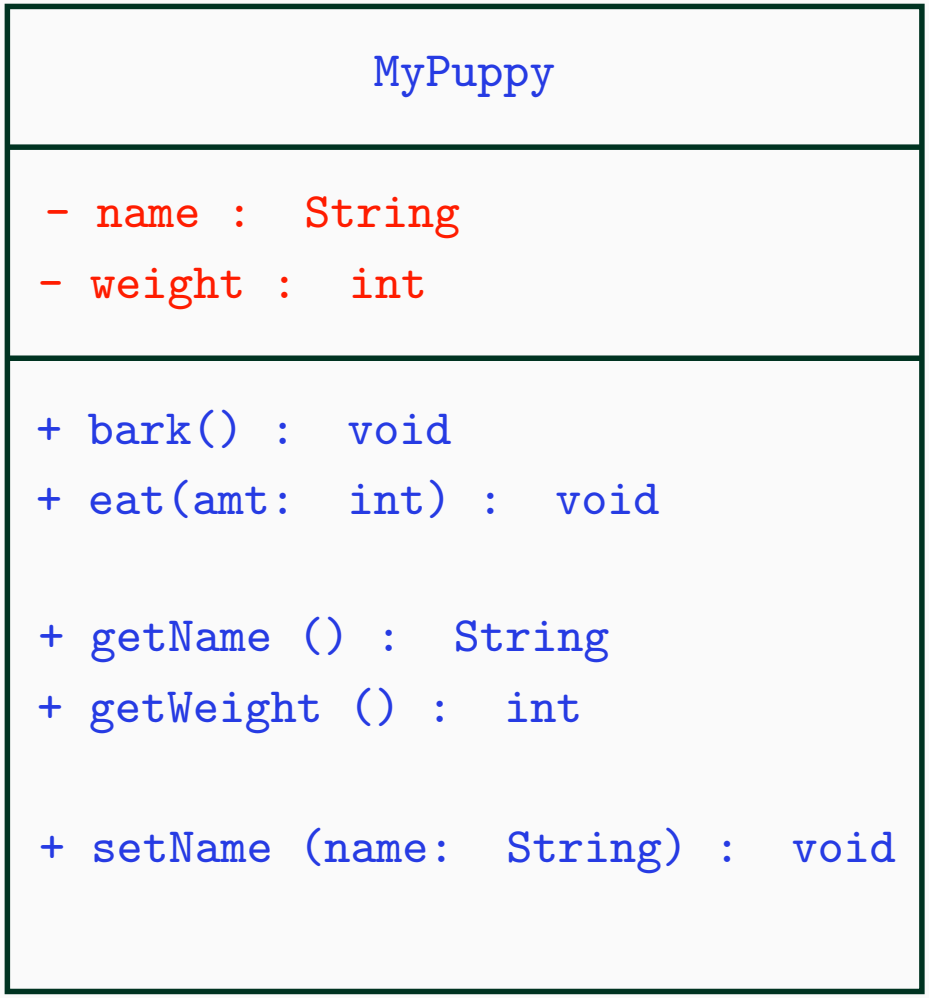


MyPuppy class (no explicit constructor)



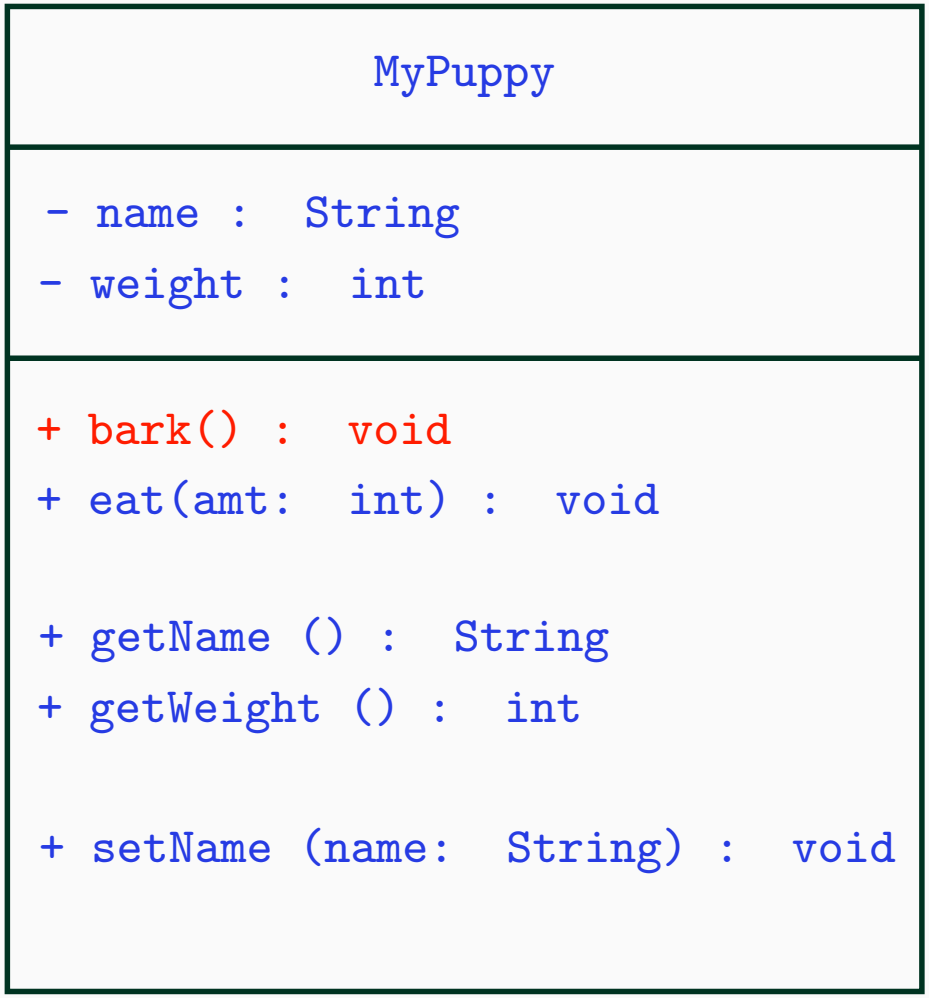

```
public class MyPuppy
{



}
```

MyPuppy class (no explicit constructor)



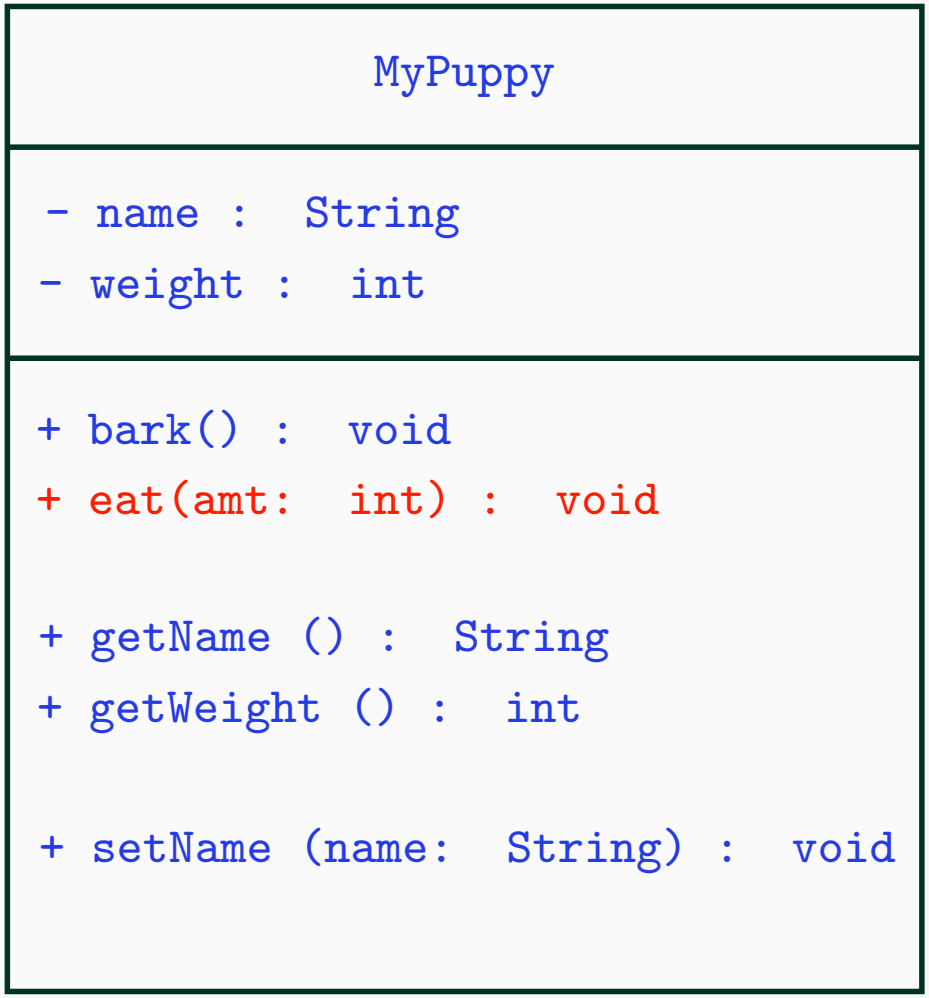

```
public class MyPuppy
{
    private String name;
    private int weight;




}
```

MyPuppy class (no explicit constructor)



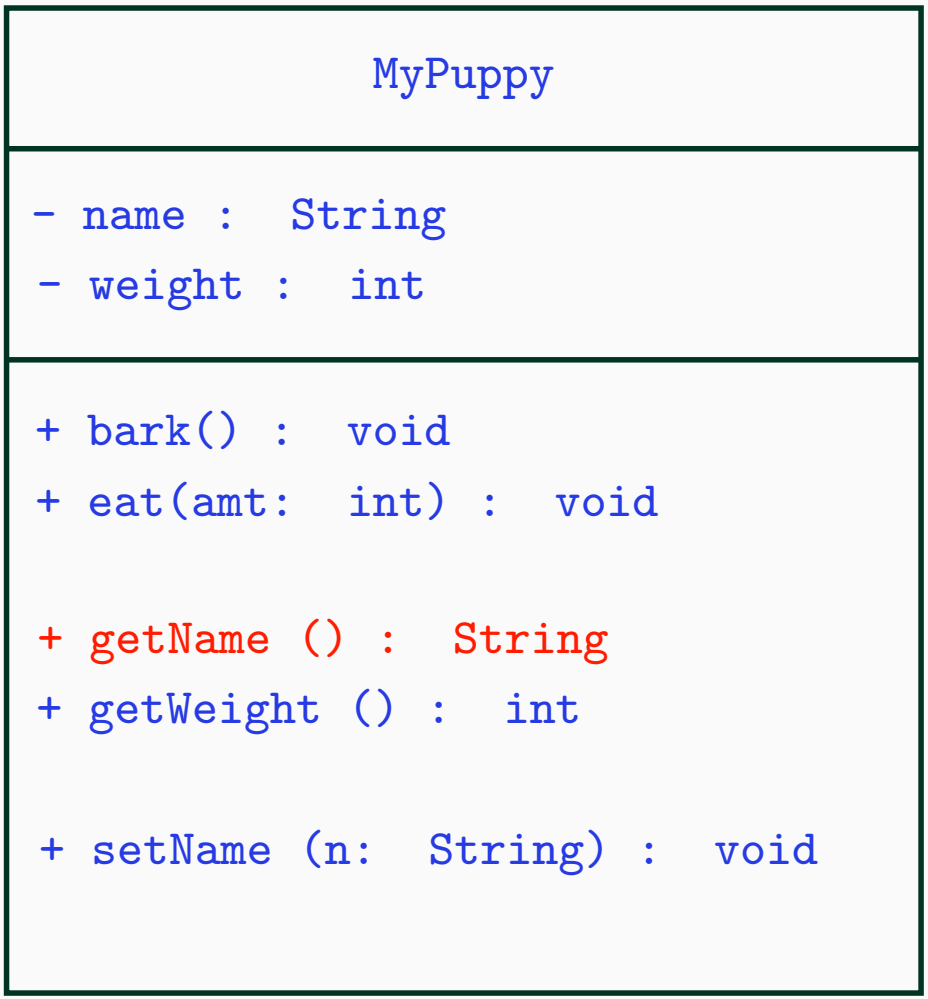

```
public class MyPuppy
{
    private String name;
    private int weight;

    public void bark ()
    {   System.out.println (name
                + "barks...arf!"
                + "...arf!\n");
    }


}
```

MyPuppy class (no explicit constructor)



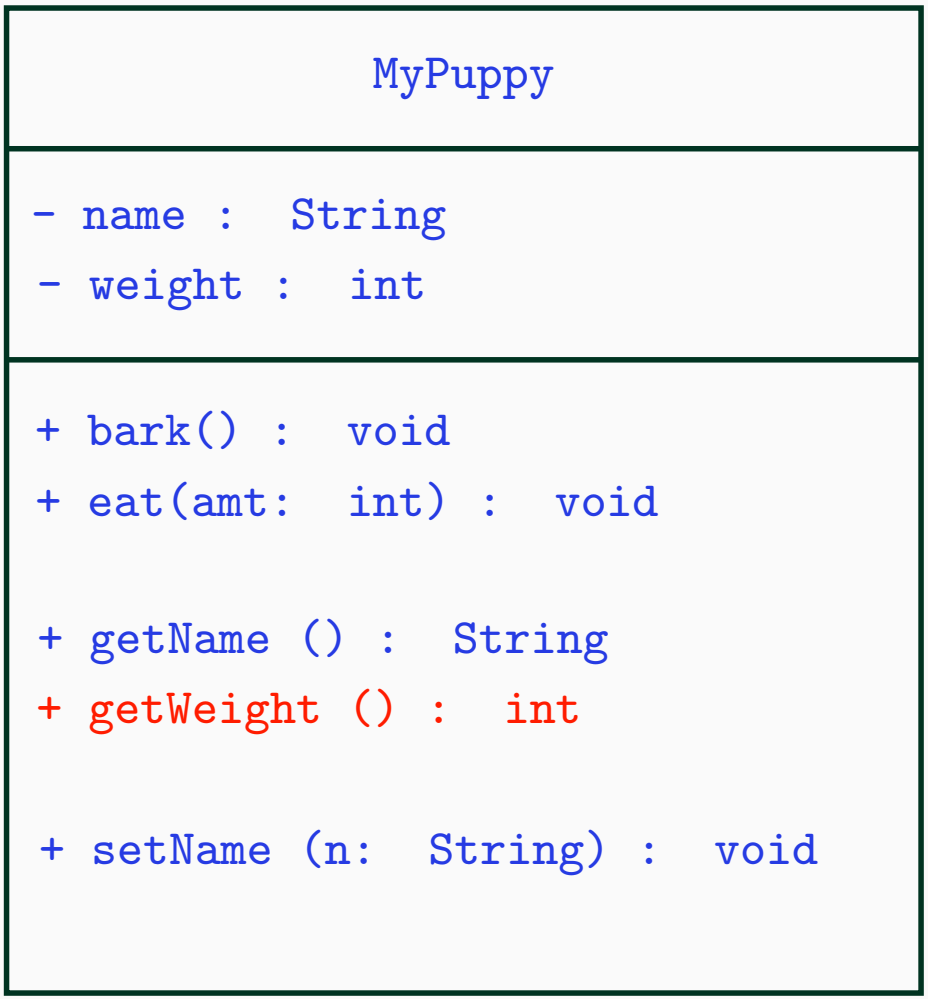

```
public class MyPuppy
{
    private String name;
    private int weight;

    public void bark ()
    {   System.out.println (name
                + "barks...arf!"
                + "...arf!\n");
    }

    public void eat (int amt)
    {   weight += amt / 2;
        System.out.println (name
                + "burps!\n");
    }
}
```

MyPuppy class (no explicit constructor)



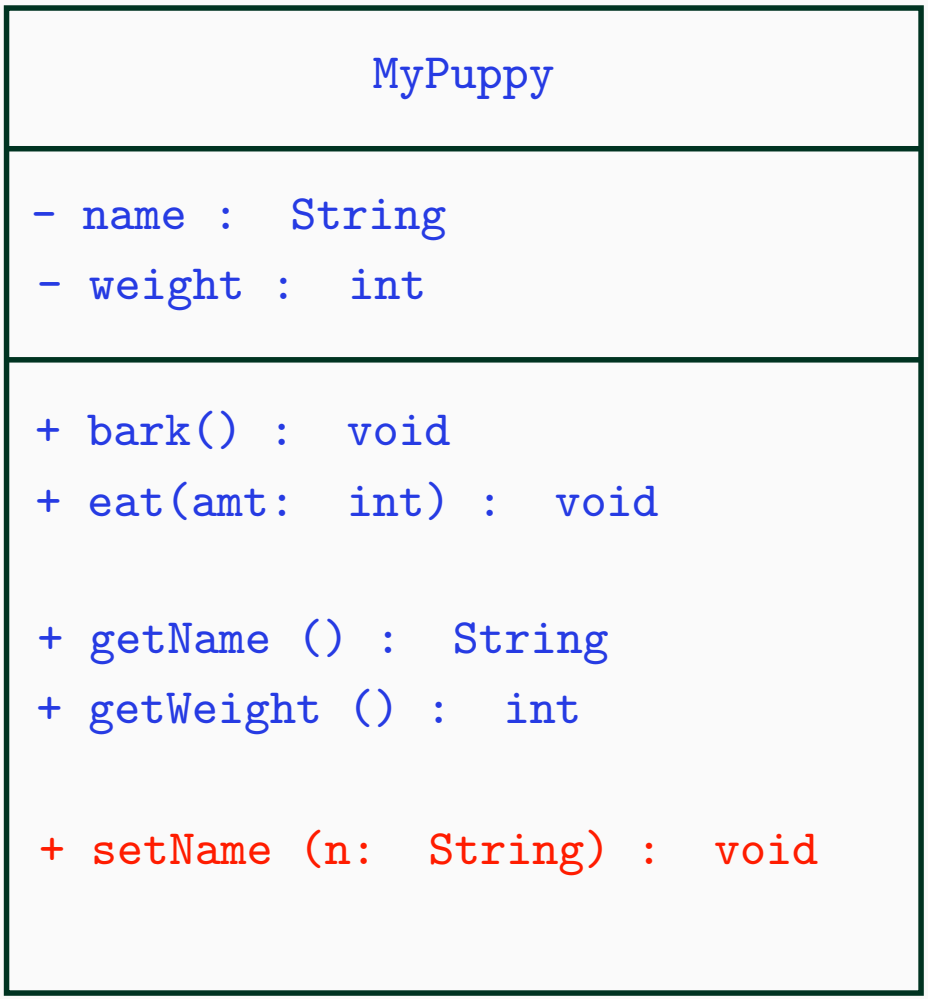

```
public class MyPuppy
{
    private String name;
    private int weight;
        :
        :
    public String getName ()
    {   return name;
    }
}
```

MyPuppy class (no explicit constructor)




```
public class MyPuppy
{
    private String name;
    private int weight;
        :
    public String getName ()
    {   return name;
    }

    public int getWeight ()
    {   return weight;
    }



}
```

MyPuppy class (no explicit constructor)




```
public class MyPuppy
{
    private String name;
    private int weight;
        :
        :
    public String getName ()
    {   return name;
    }

    public int getWeight ()
    {   return weight;
    }

    public void setName (String n)
    {   name = n;
    }
}
```

`MyPuppy` driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

`MyPuppy` driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

Create driver class for to test `MyPuppy`.

`MyPuppy` driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

Create driver class for to test `MyPuppy`.

1. Create the first puppy. ▸ instantiation

`MyPuppy` driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

Create driver class for to test `MyPuppy`.

1. Create the first puppy. ▸ instantiation

```
MyPuppy dog;
dog = new MyPuppy ();
```

`MyPuppy` driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

Create driver class for to test `MyPuppy`.

1. Create the first puppy. ▸ instantiation

```
MyPuppy dog;
dog = new MyPuppy ();
```

2. Display the puppy's name and weight.

MyPuppy driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

Create driver class for to test `MyPuppy`.

1. Create the first puppy. ▸ instantiation

```
MyPuppy dog;
dog = new MyPuppy ();
```

2. Display the puppy's name and weight.

```
System.out.println (getName());
System.out.println (getWeight());
```

MyPuppy driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

Create driver class for to test `MyPuppy`.

1. Create the first puppy. ▸ instantiation

```
MyPuppy dog;
dog = new MyPuppy ();
```

2. Display the puppy's name and weight.

```
System.out.println (getName());
System.out.println (getWeight());
```

MyPuppy driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

Create driver class for to test `MyPuppy`.

1. Create the first puppy. ▸ instantiation

```
MyPuppy dog;
dog = new MyPuppy ();
```

2. Display the puppy's name and weight.

```
System.out.println (dog.getName());
System.out.println (dog.getWeight());
```

MyPuppy driver class

A *driver class* is just a class that contains a `main()` method. It is used to test the classes you've created.

Create driver class for to test `MyPuppy`.

1. Create the first puppy. ▸ instantiation

```
MyPuppy dog;
dog = new MyPuppy ();
```

2. Display the puppy's name and weight.

```
System.out.println (dog.getName());
System.out.println (dog.getWeight());
```

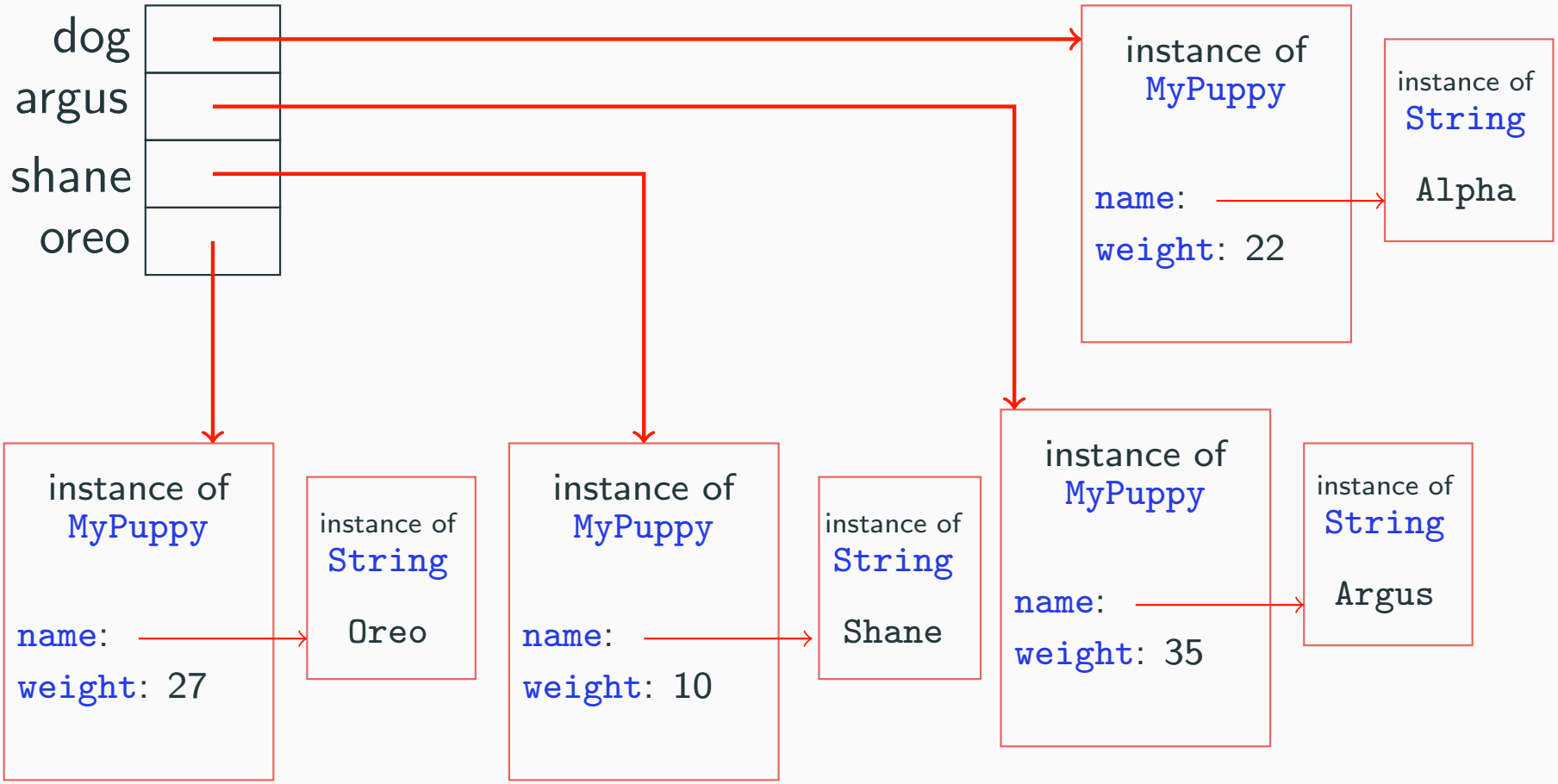
3. Name this puppy Alpha. Alpha eats 5 units of dog food. Alpha barks. Display Alpha's current weight.

MyPuppy driver class

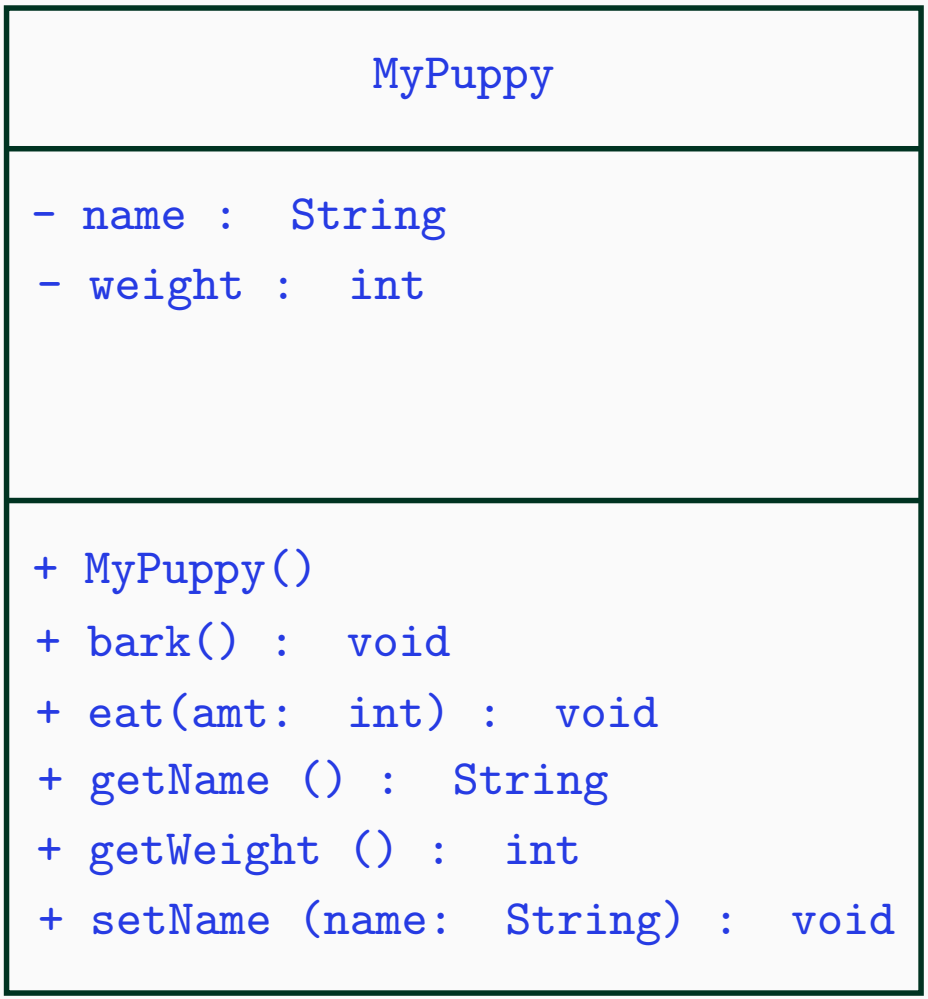
Now...

1. Create three more puppies: Argus, Shane, and Oreo.
2. It's feeding time again!
    - Alpha eats 20 units of dog food.
    - Argus eats 50 units of dog food.
    - Oreo eats 35 units of dog food.
    - Shane is on diet, and will just eat later.
3. Display the current weight of the puppies.

Alpha, Argus, Oreo, and Shane

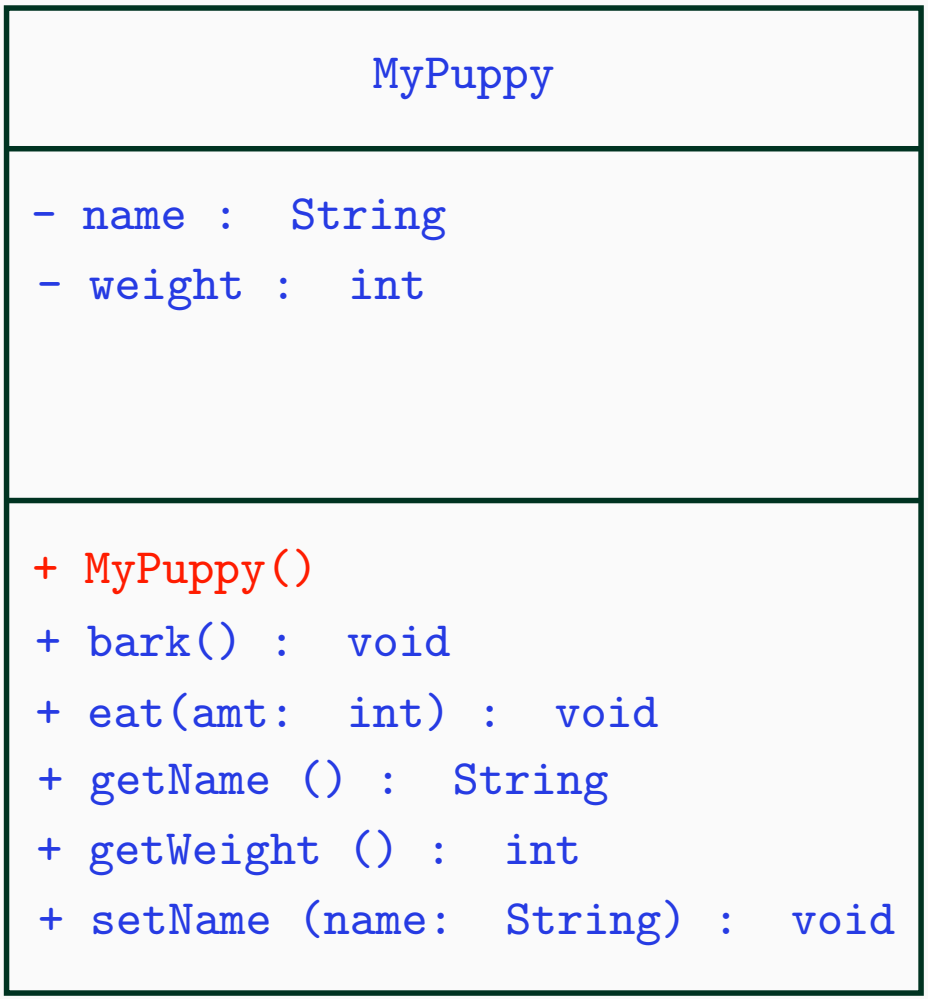


MyPuppy class (with constructor)



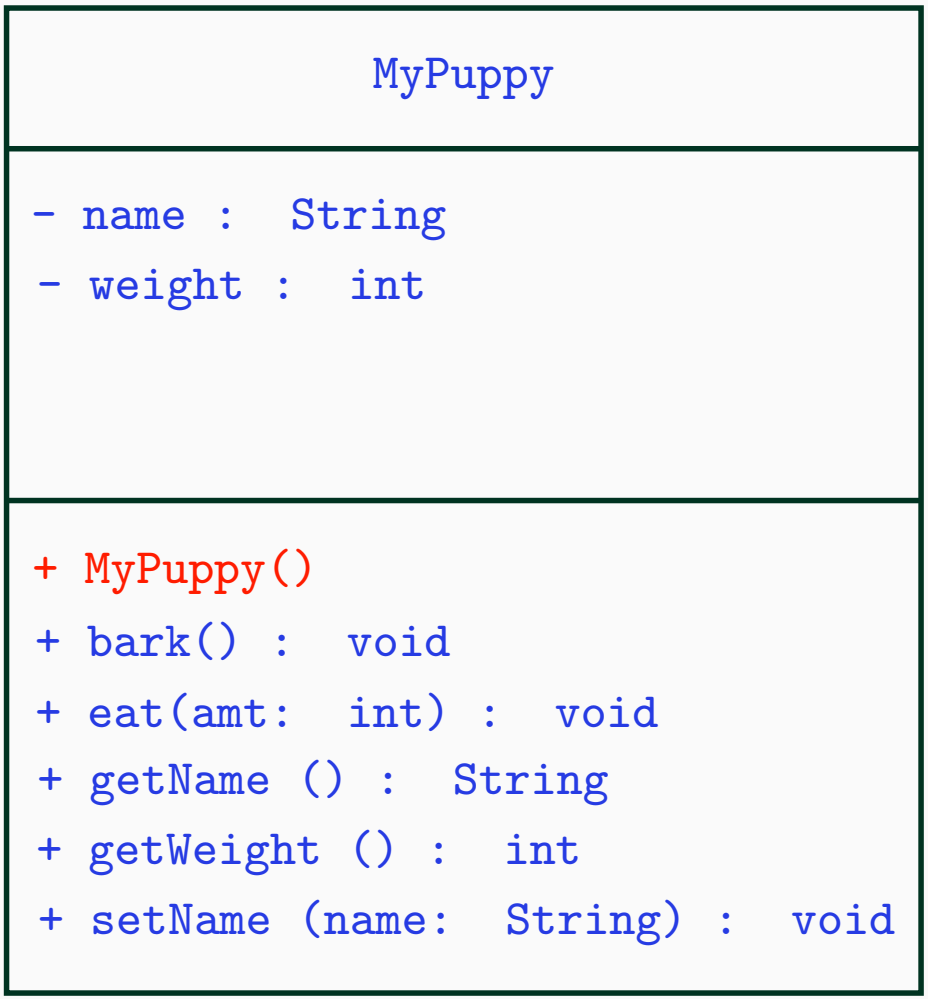

```
public class MyPuppy
{
    private String name;
    private int weight;



}
```

MyPuppy class (with constructor)



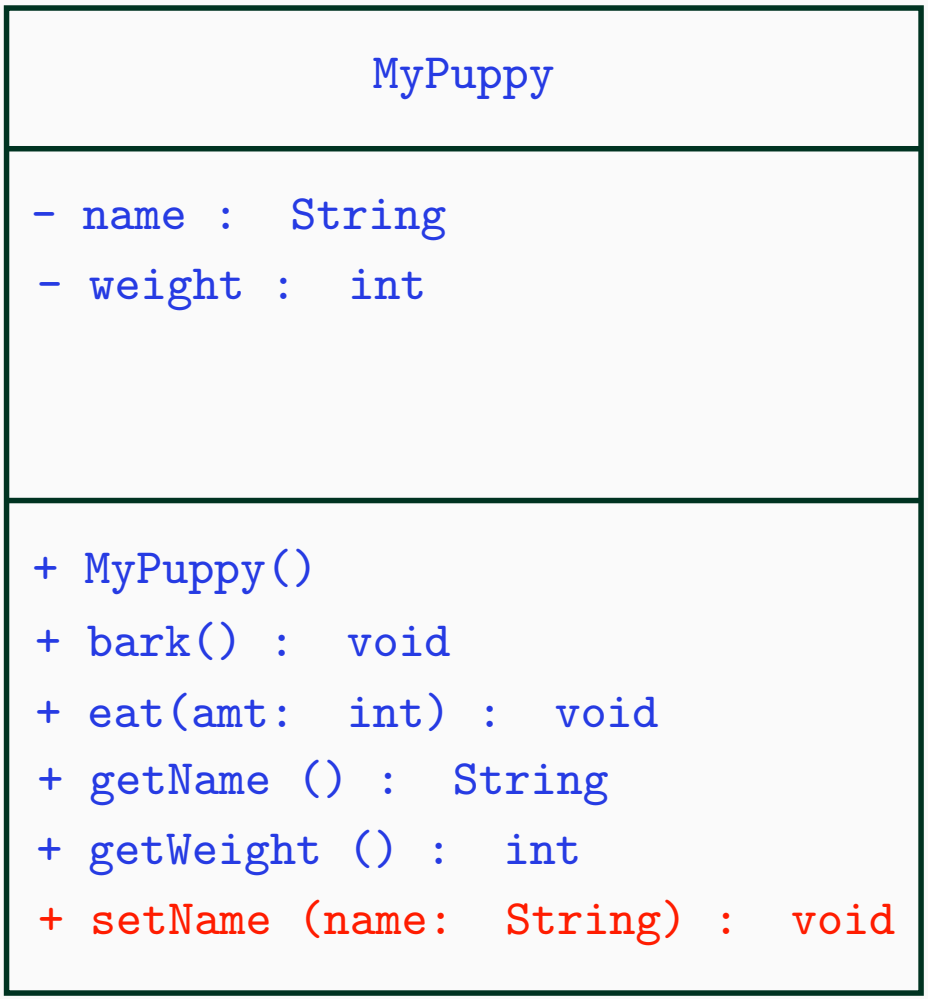

```
public class MyPuppy
{
    private String name;
    private int weight;

    public MyPuppy ()
    {


    }



}
```

MyPuppy class (with constructor)



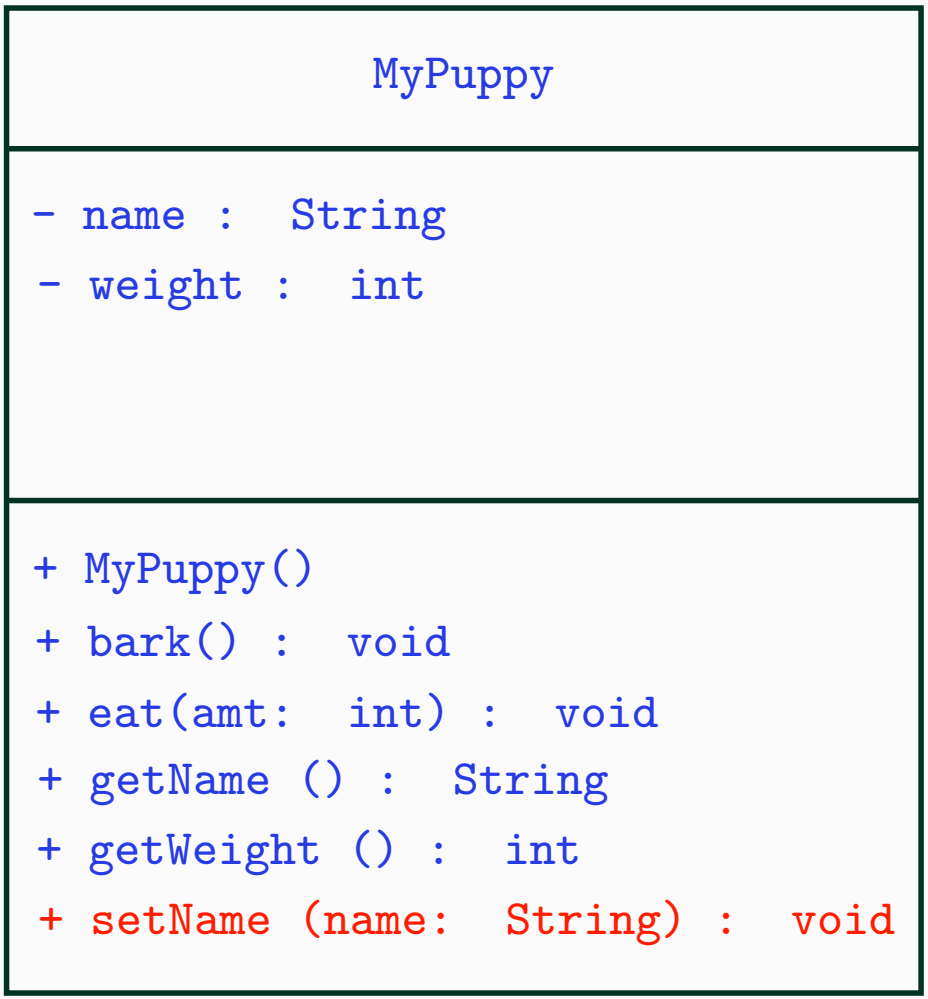

```
public class MyPuppy
{
    private String name;
    private int weight;

    public MyPuppy ()
    {
        name = "puppy";
        weight = 10;
    }




}
```

MyPuppy class (with constructor)



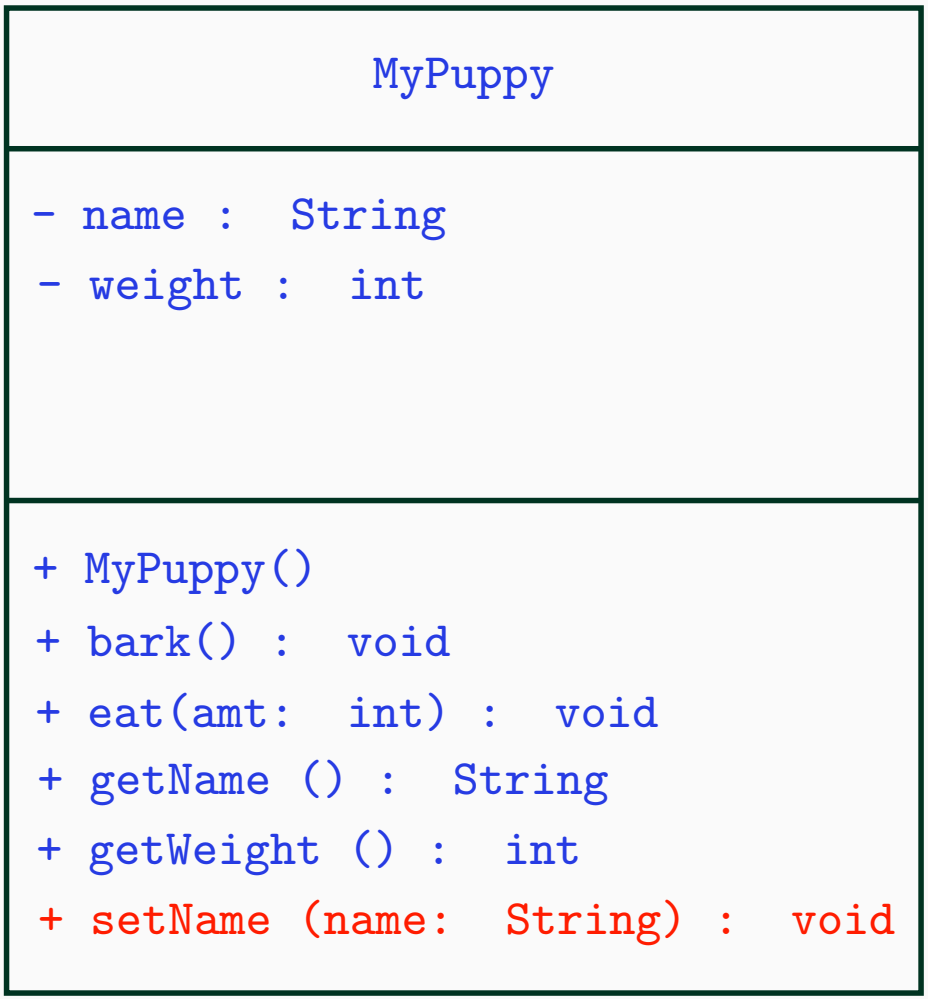

```
public class MyPuppy
{
    private String name;
    private int weight;

    public MyPuppy ()
    {
        name = "puppy";
        weight = 10;
    }

    public void setName (String name)
    {

    }
}
```

MyPuppy class (with constructor)




```
public class MyPuppy
{
    private String name;
    private int weight;

    public MyPuppy ()
    {
        name = "puppy";
        weight = 10;
    }

    public void setName (String name)
    {

    }
}
```

MyPuppy class (with constructor)




```
public class MyPuppy
{
    private String name;
    private int weight;

    public MyPuppy ()
    {
        name = "puppy";
        weight = 10;
    }

    public void setName (String name)
    {
        this.name = name;
    }
}
```

The keyword `this`

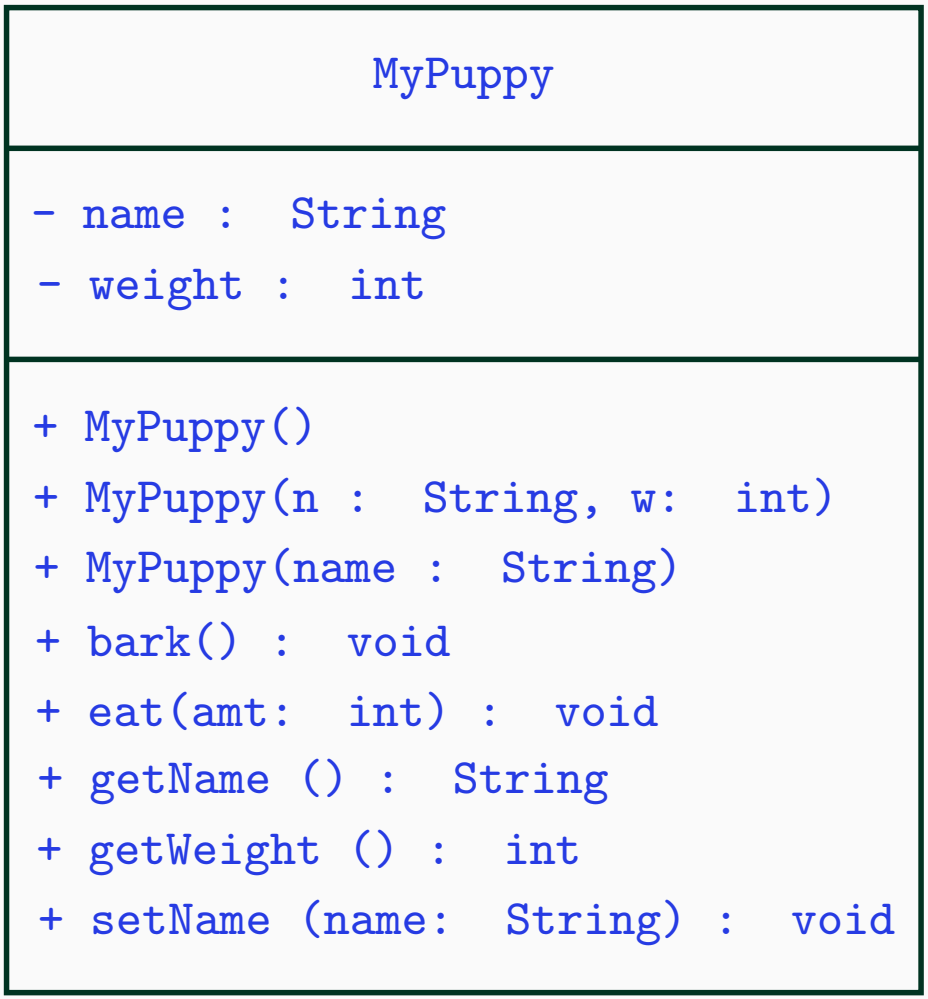
- used to refer to *this* instance inside a class definition.
- resolves naming conflict between an attribute (member variable) and a parameter of the method
- when the statement `return this;` is used inside a method, that method returns *this* instance to the caller.

Activity

After updating your `MyPuppy` class, compile and run the driver class.

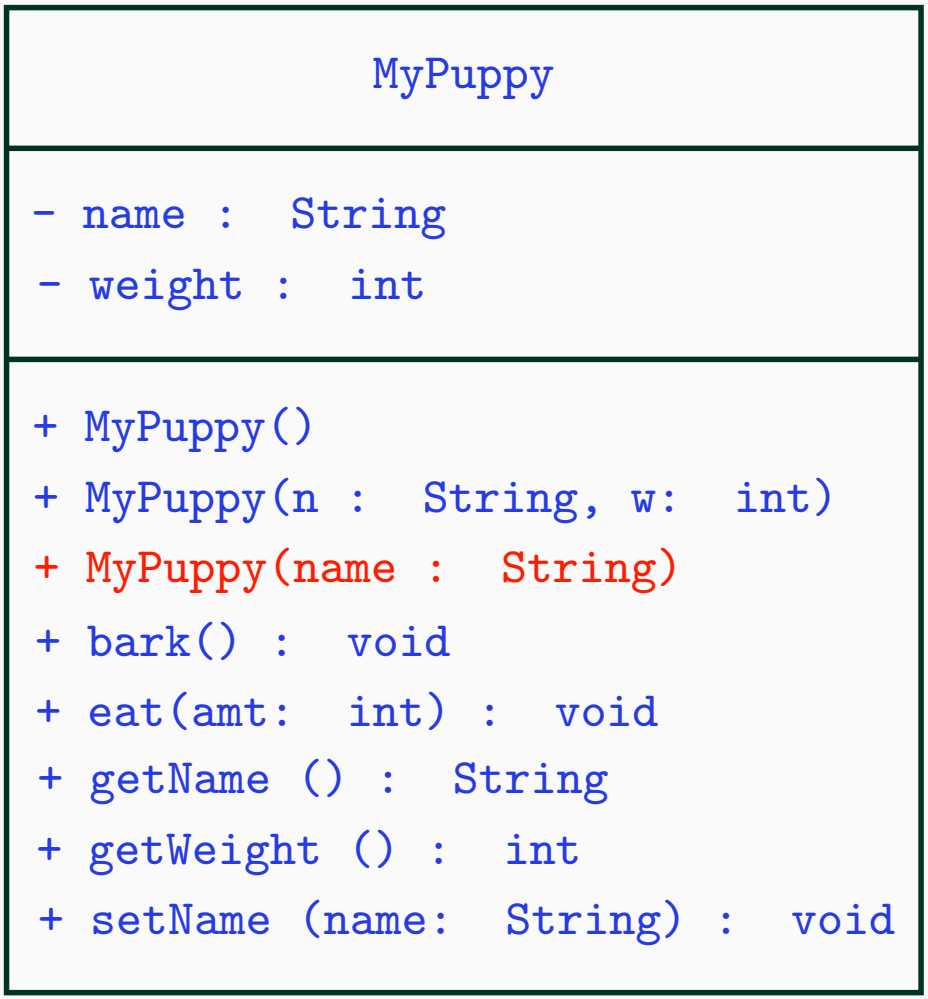
How is the output now different from the output earlier?

MyPuppy class (with overloaded constructors)



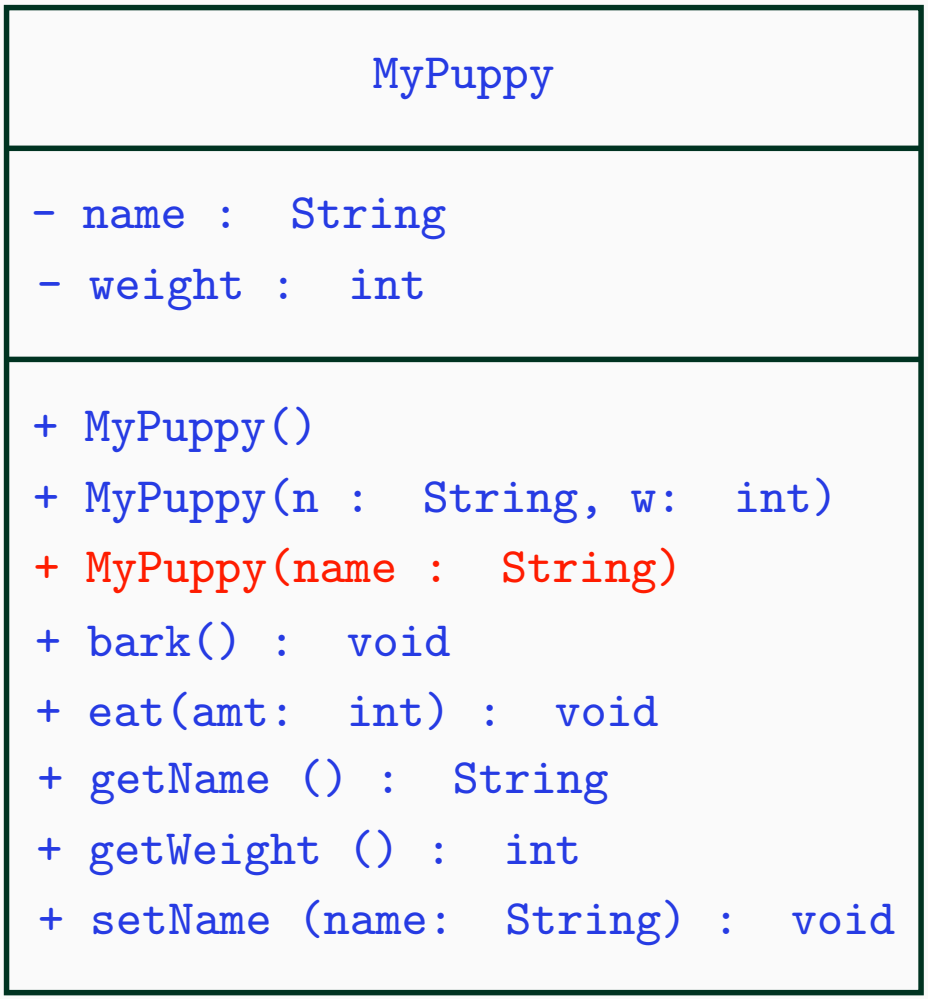

```
public class MyPuppy
{   private String name;
    private int weight;

    public MyPuppy ()
    {   name = "puppy";
        weight = 10;
    }




}
```

MyPuppy class (with overloaded constructors)



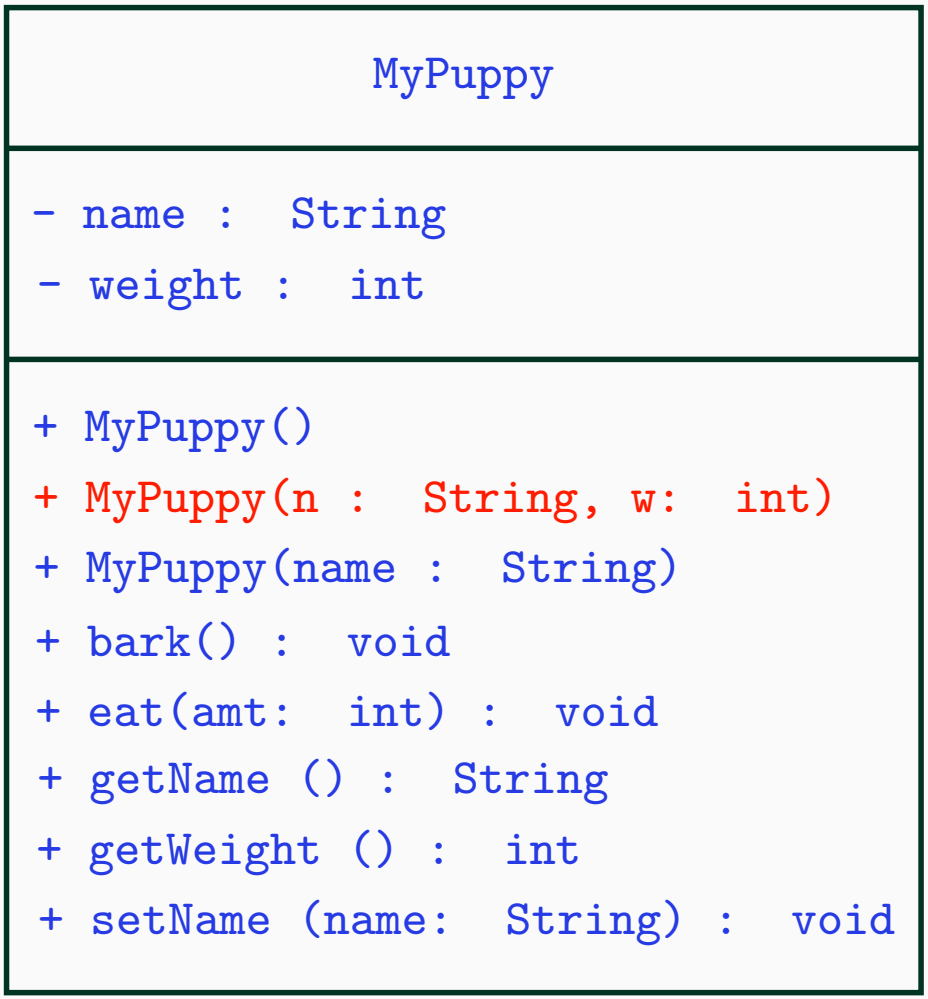

```
public class MyPuppy
{   private String name;
    private int weight;

    public MyPuppy ()
    {   name = "puppy";
        weight = 10;
    }

    public MyPuppy (String name)
    {

    }



}
```

MyPuppy class (with overloaded constructors)



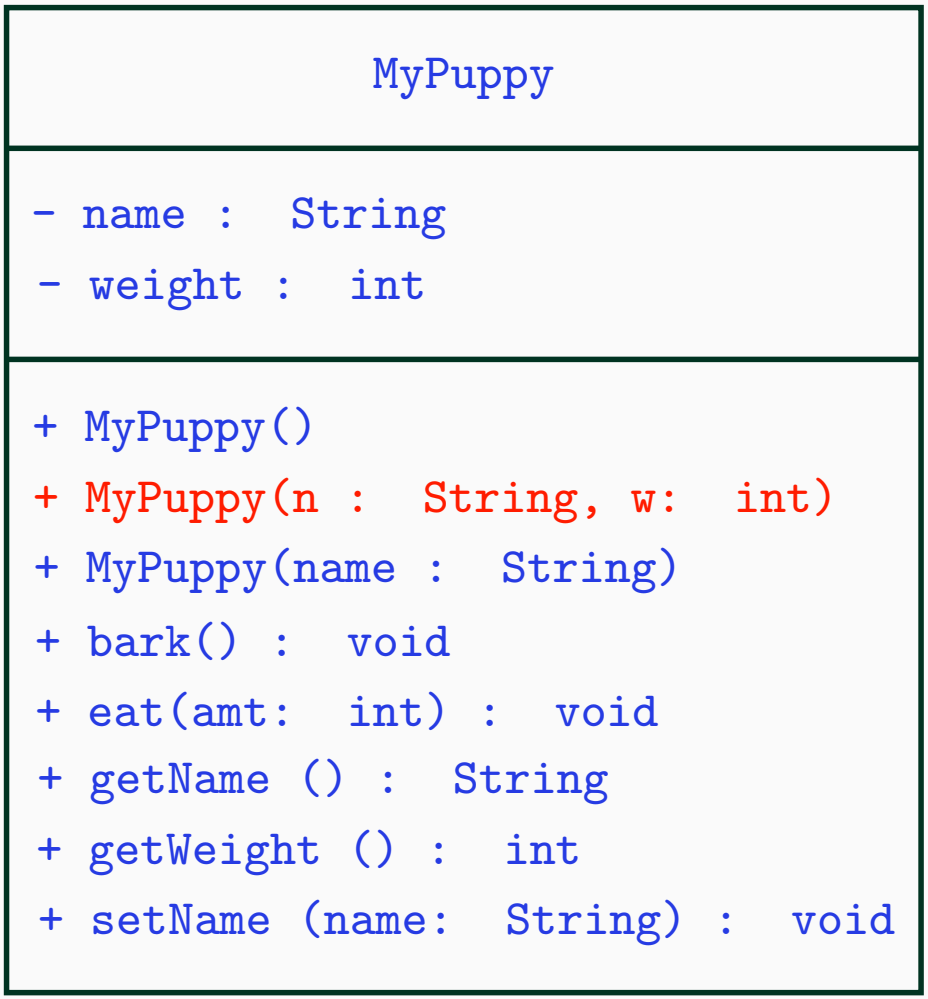

```
public class MyPuppy
{   private String name;
    private int weight;

    public MyPuppy ()
    {   name = "puppy";
        weight = 10;
    }

    public MyPuppy (String name)
    {   this ();
        this.name = name;
    }


}
```

MyPuppy class (with overloaded constructors)




```
public class MyPuppy
{   private String name;
    private int weight;

    public MyPuppy ()
    {   name = "puppy";
        weight = 10;
    }

    public MyPuppy (String name)
    {   this ();
        this.name = name;
    }

    public MyPuppy (String n, int w)
    {

    }
}
```

MyPuppy class (with overloaded constructors)




```
public class MyPuppy
{   private String name;
    private int weight;

    public MyPuppy ()
    {   name = "puppy";
        weight = 10;
    }

    public MyPuppy (String name)
    {   this ();
        this.name = name;
    }

    public MyPuppy (String n, int w)
    {   name = n;
        this.weight = w;
    }
}
```

this()

- used to call another constructor in the same class. This is what is called *explicit constructor invocation*.
- if used, must always be the first line in the constructor.

Activity

After updating your `MyPuppy` class, update your driver class.

Update how the three puppies Argus, Shane, and Oreo are created.

- Argus is the puppy's name when created.
- Shane's weight is 30 when created.
- Oreo's name was originally Mocha, and has a weight of 25 (when it was created).

Compile and run your program.

`Date` class

Code your `Date` class.

☺ **Thank you!** ☺

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.
  1. declare a variable of type `MyPuppy`.

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.
  1. declare a variable of type `MyPuppy`.

```
MyPuppy dog;
```

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.
  1. declare a variable of type `MyPuppy`.

     ```
     MyPuppy dog;
     ```

  2. instantiate a `MyPuppy` object

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.
  1. declare a variable of type `MyPuppy`.

     ```
     MyPuppy dog;
     ```

  2. instantiate a `MyPuppy` object

     ```
     dog = new MyPuppy ();
     ```

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.
  1. declare a variable of type `MyPuppy`.

     ```
     MyPuppy dog;
     ```

  2. instantiate a `MyPuppy` object

     ```
     dog = new MyPuppy ();
     ```

- `new` is always followed by a call to a constructor.

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.
  1. declare a variable of type `MyPuppy`.

     ```
     MyPuppy dog;
     ```

  2. instantiate a `MyPuppy` object

     ```
     dog = new MyPuppy ();
     ```

- `new` is always followed by a call to a constructor.
- Recall: Java automatically generates a default constructor, if no constructor is declared.

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.
  1. declare a variable of type `MyPuppy`.

     ```
     MyPuppy dog;
     ```

  2. instantiate a `MyPuppy` object

     ```
     dog = new MyPuppy ();
     ```

- `new` is always followed by a call to a constructor.
- Recall: Java automatically generates a default constructor, if no constructor is declared.
- Recall: The default constructor is a no parameter constructor.

Instantiation

- Recall: A class is just a blueprint. The realization of this blueprint is the object of this class.
- We create an object of this class through *instantiation*.
- To instantiate, the keyword `new` is used.
  1. declare a variable of type `MyPuppy`.

     ```
     MyPuppy dog;
     ```

  2. instantiate a `MyPuppy` object

     ```
     dog = new MyPuppy ();
     ```

- `new` is always followed by a call to a constructor.
- Recall: Java automatically generates a default constructor, if no constructor is declared.
- Recall: The default constructor is a no parameter constructor.

◂ Back