Unit 1 - Lesson 8
Inheritance

Computer Science A

Warm Up

● ○ ○

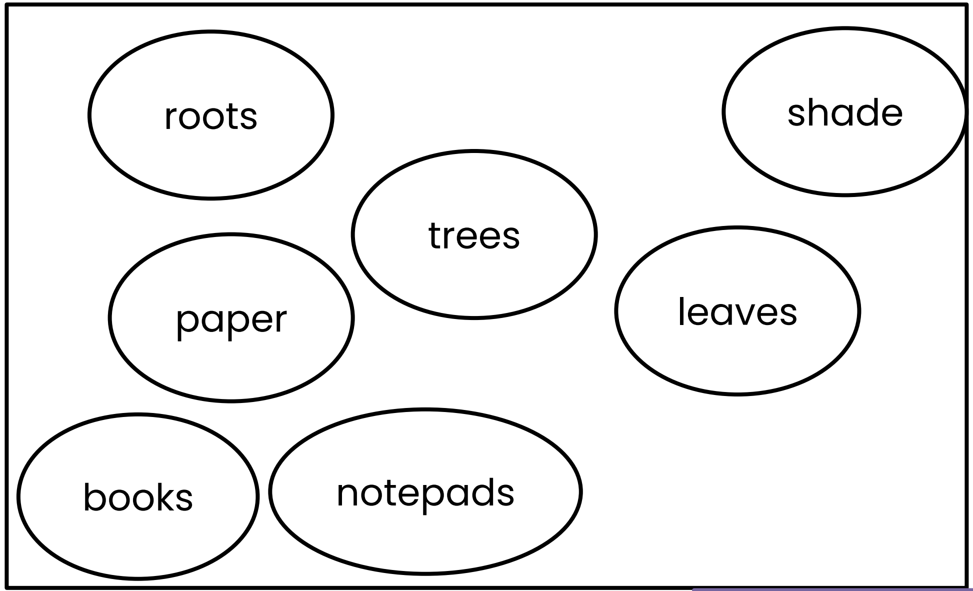
What is object-oriented programming?

✅ Do This:

Create a **concept map** to answer the question **"What is object-oriented programming?"**

Write any **concepts** or **ideas** that come to mind.

Example: What is a tree?



📝 **Unit 1 Guide**

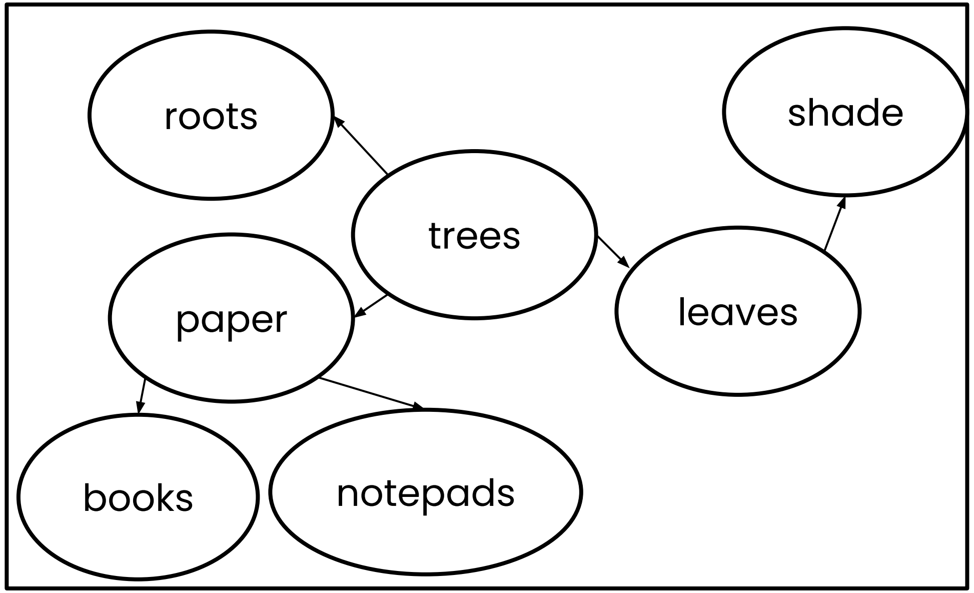
How are these concepts connected?

✅ Do This:

Review the **concepts** and **ideas** you came up with.

Draw **arrows** to **connect** the concepts.

Example: What is a tree?

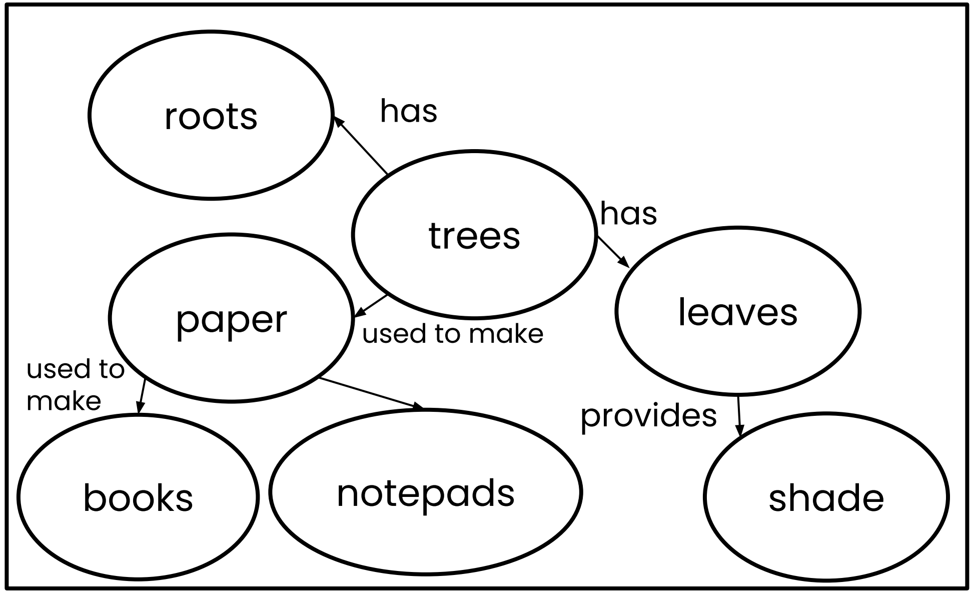


How are these concepts connected?

✅ **Do This:**

Label the arrows with **verbs** or **short descriptions** to identify the **relationships** between the concepts and ideas.

Example: What is a tree?



✅ Do This:

Share your concept map with a neighbor.

Compare the concepts and relationships you wrote.

Add to or **revise** your concept map based on your discussion.

Activity

● ● ○

🎯 Lesson Objectives

By the end of this lesson, you will be able to . . .

- Explain the purpose and functionality of inheritance
- Identify use cases for creating subclasses of an existing class
- Write a subclass that extends a superclass

🚀 Question of the Day

Why would I use inheritance?

Painter
xLocation yLocation direction remainingPaint

turnLeft()
move()
paint(color)
takePaint()
canMove()
isOnBucket()
hasPaint()

💡 Discuss:

If you could add a new method to the **`Painter`** class, what method would you want to add?

✅ Do This:

The software engineer knows that this `Instrument` class works to **represent a guitar** at a music store, but the owner wants to also sell **pianos**.

How should the software engineer implement these changes?

What if the music store owner later decides to also sell **other instruments**?

`Instrument`
`model` `numStrings`

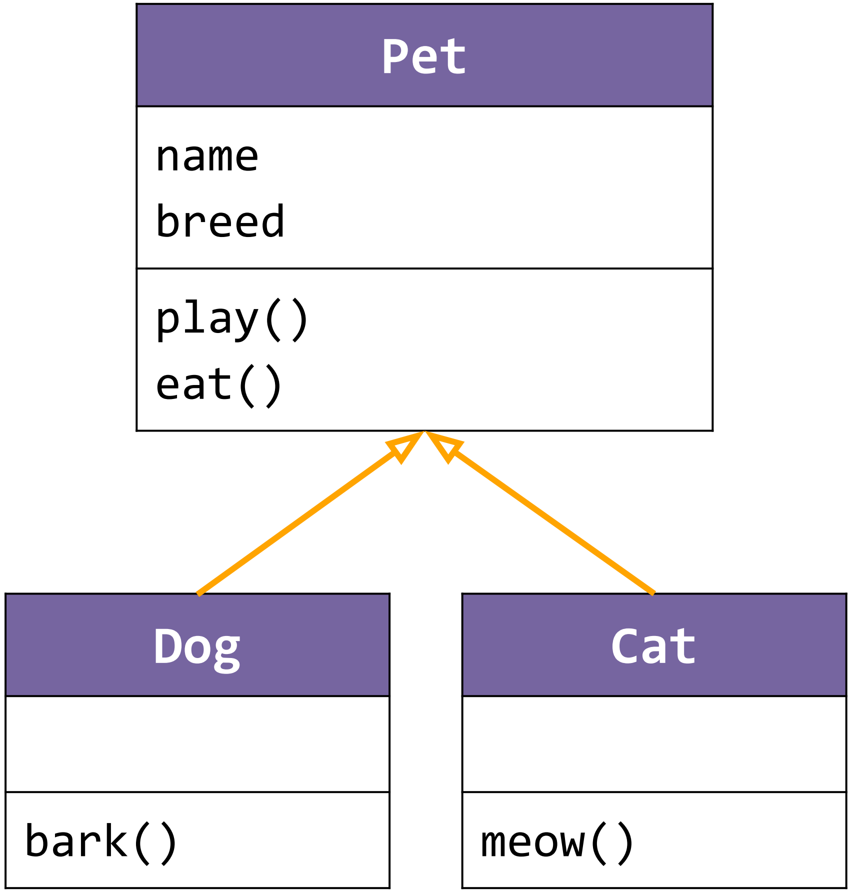
`tune()`
`restring()`
`play()`

🎥 Inheritance

What is inheritance?

Complete the guided notes on the 📝 **Unit 1 Guide**.

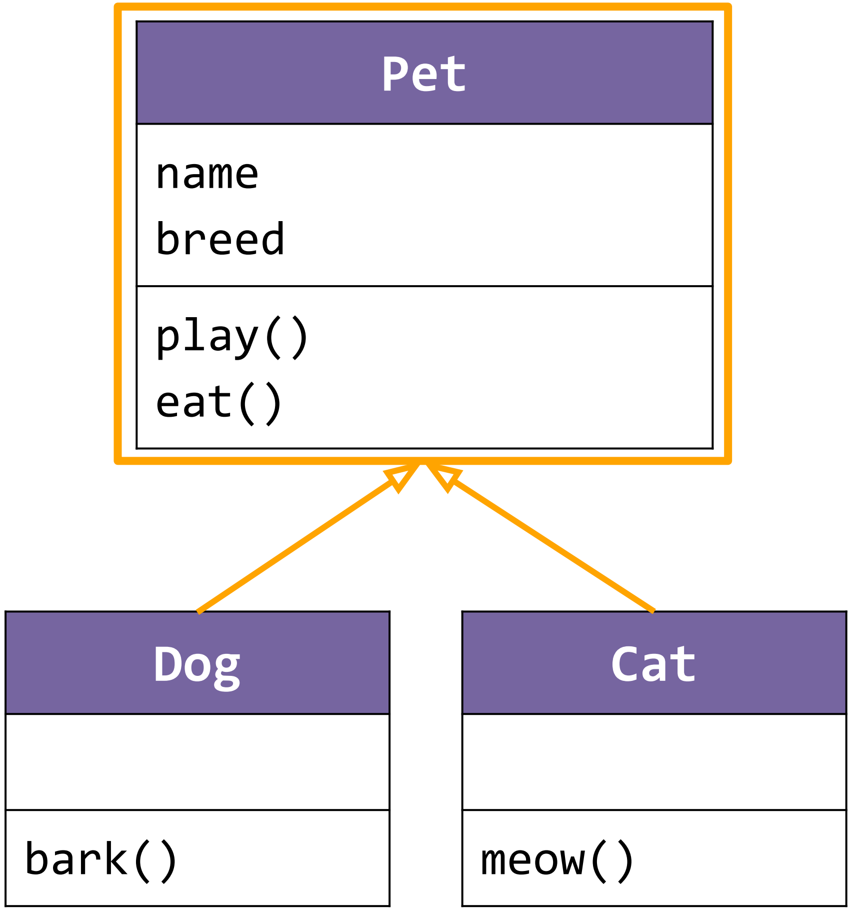
Inheritance is an object-oriented programming principle where a **subclass inherits** the **attributes** and **behaviors** of a **superclass**.



📝 **Unit 1 Guide**

A **superclass** is a class that **can be extended** to create subclasses.

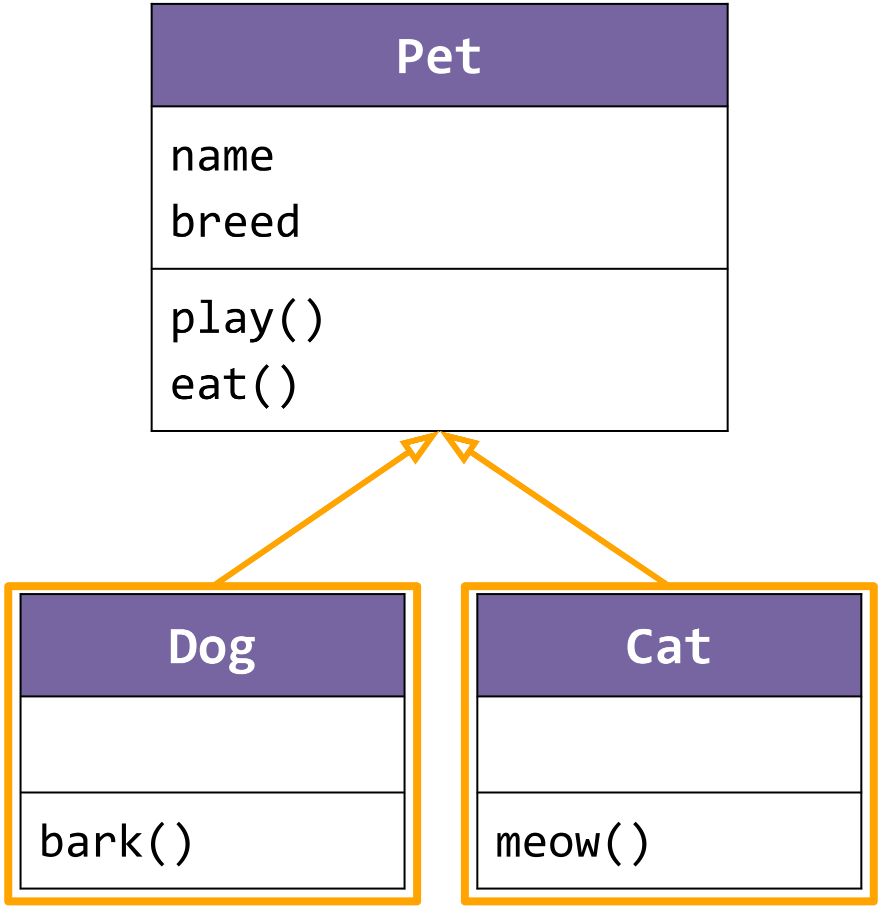
It **contains** the **attributes** and **behaviors** that can be **shared** with subclasses.



📝 **Unit 1 Guide**

A subclass is a class that **extends a superclass** and **inherits** its **attributes** and **behaviors**.

It has the **same attributes** and **behaviors** as the **superclass** and can have **additional ones** of its own.

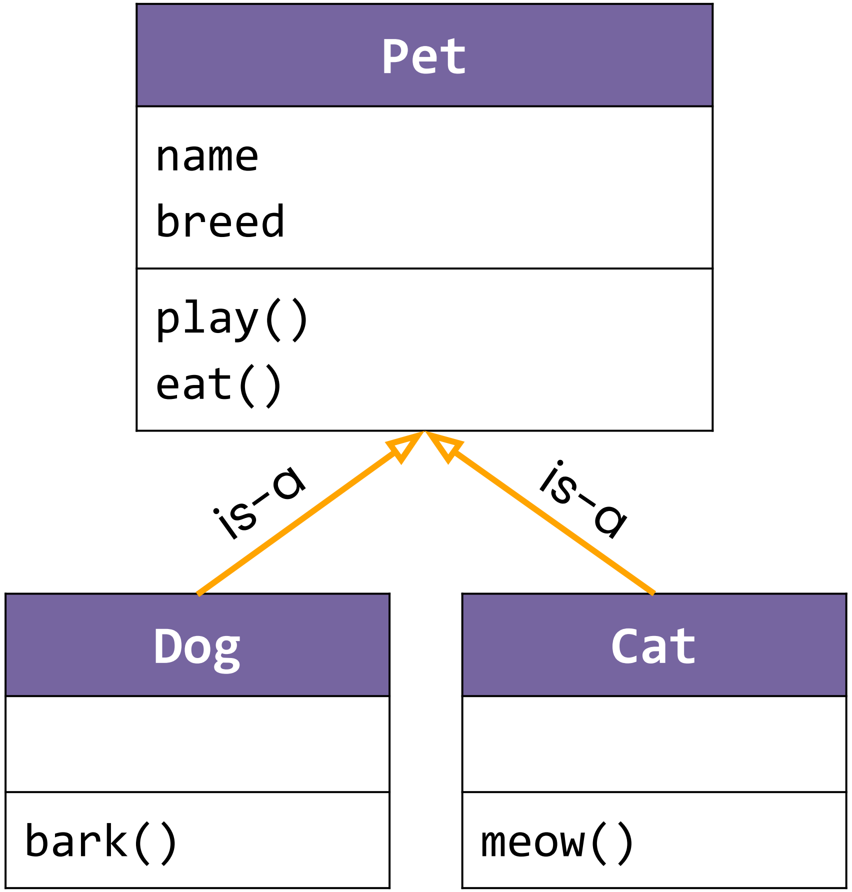


📝 **Unit 1 Guide**

Inheritance creates an **is-a relationship** between the superclass and its subclasses.

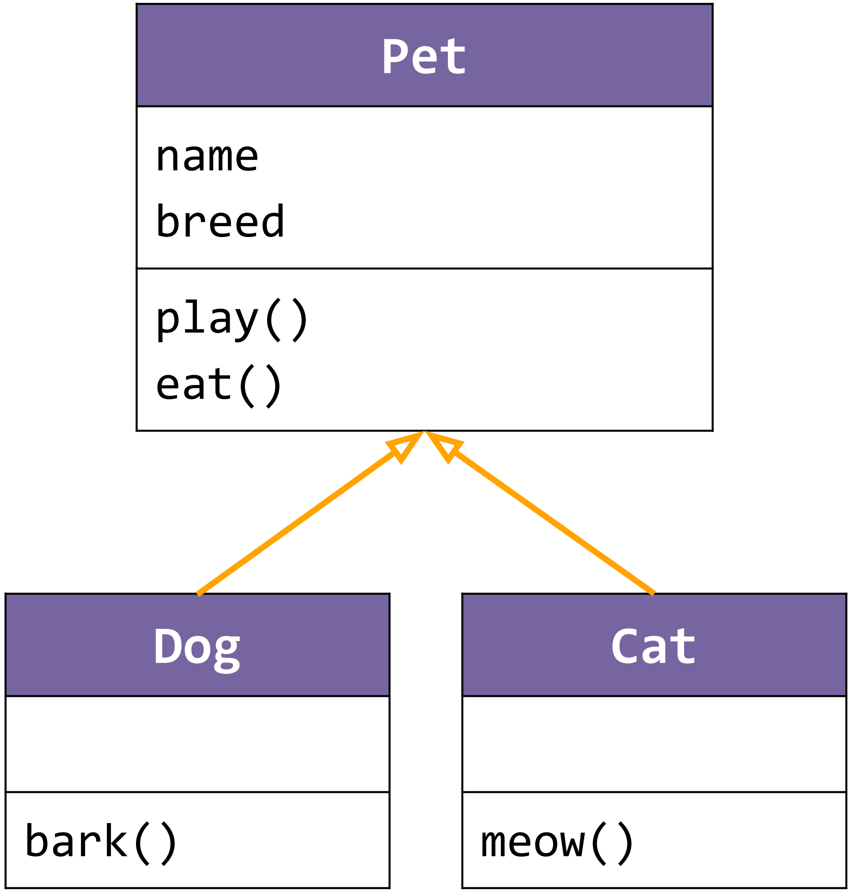
A **Dog is-a Pet.**

A **Cat is-a Pet.**



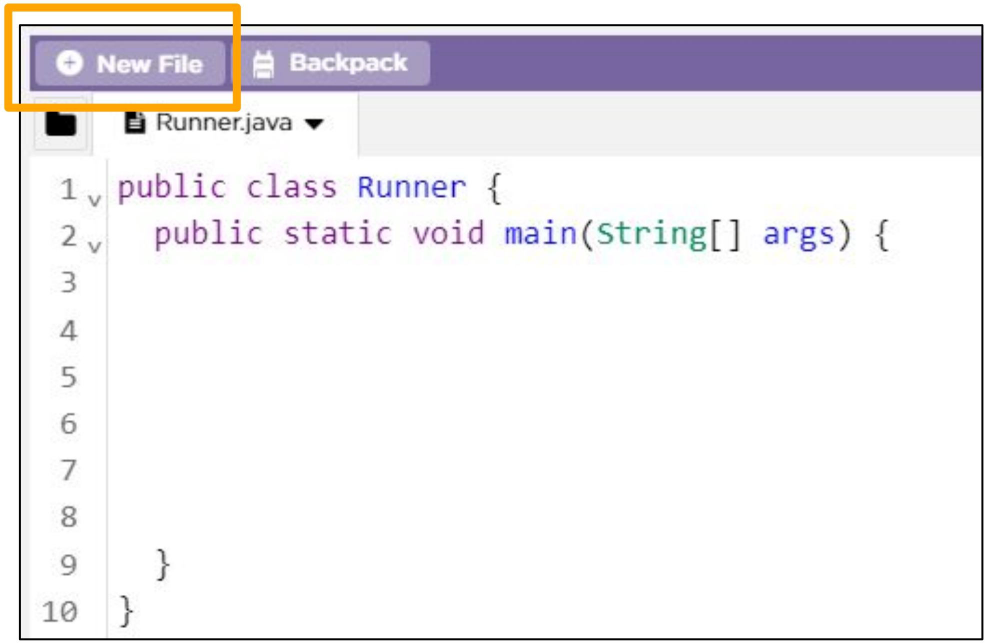
💡 Discuss:

What are some scenarios where you would want to use inheritance?



Creating a New Class

To create a **new class** in **Java Lab**, we start by creating a **new file**.

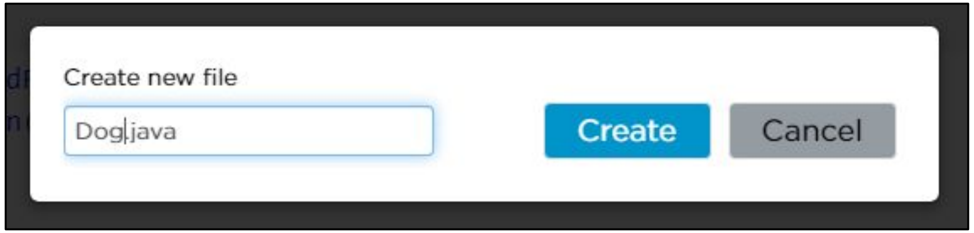


```
public class Runner {
  public static void main(String[] args) {






  }
}
```

Creating a New Class

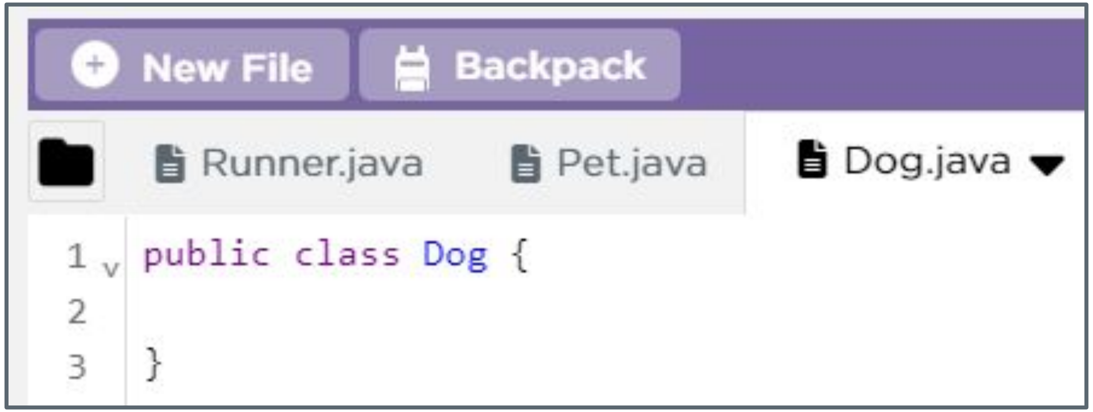
We then enter the **name** of the class we want to create.

Make sure the new file ends in `.java`!



Creating a New Class

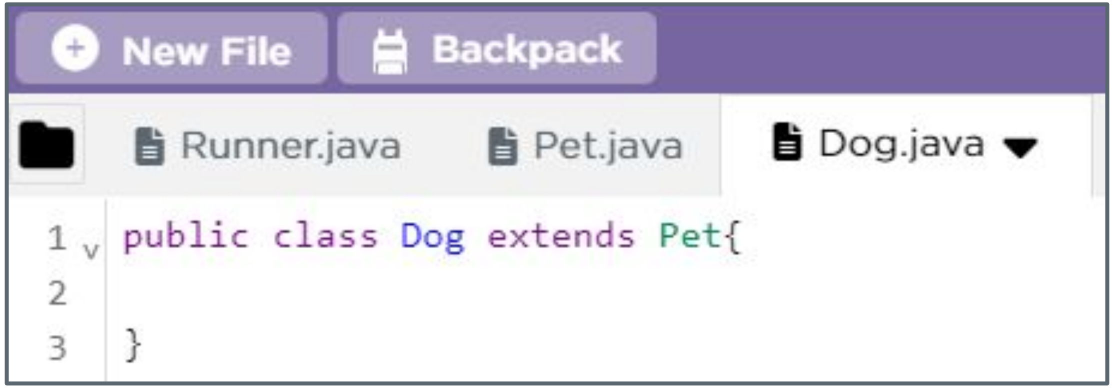
In our new class, we write the **class header**.



```
public class Dog {

}
```

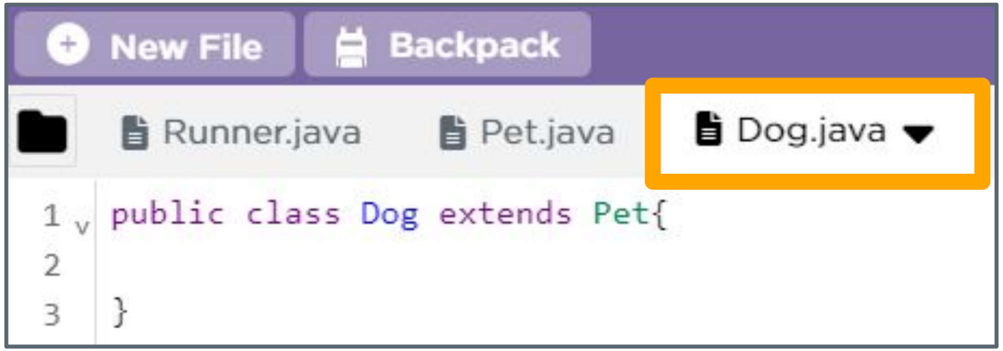
Creating a New Class

We add the **extends keyword** to the **class header** to indicate that **this is a subclass** and the **superclass that it inherits from**.



```java
public class Dog extends Pet{

}
```

Our Java programs now consist of two types of classes.



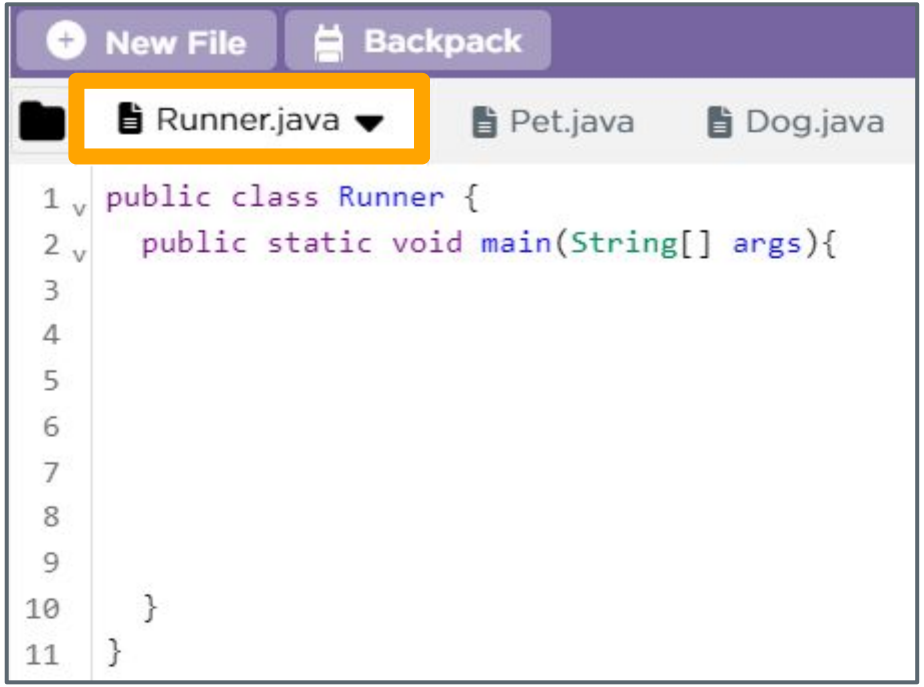
```
public class Dog extends Pet{

}
```

The **class that represents an object** and **contains its attributes and behaviors**.

We can **instantiate objects** from this class.

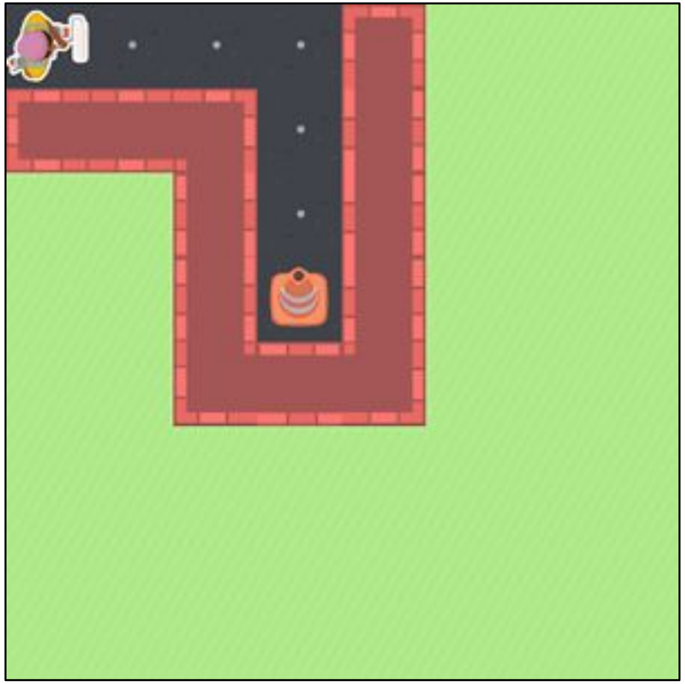
Our Java programs now consist of two types of classes.

The *tester class* , which is the **class that contains the** `main` **method** and from **where the program starts running**.



```
public class Runner {
  public static void main(String[] args){









  }
}
```

🎉Practice



Navigate to Lesson 8, Level 1

✅ Do This:

1. **Level 1 –** Check for Understanding
2. **Level 2 –** Create your `PainterPlus` class and instantiate a `PainterPlus` object
3. **Level 3 –** Use your `PainterPlus` object to reach the traffic cone

💡 Discuss:

What happened when you called **`Painter`** class methods?

Why do you think this happened?

Wrap Up

Painter
xLocation yLocation direction remainingPaint

turnLeft()
move()
paint(color)
takePaint()
canMove()
isOnBucket()
hasPaint()

💡 Discuss:

What are some things you want a **`PainterPlus`** object to do that a **`Painter`** object **can't do**?

🎉 Today, you learned about . . .

- The purpose and functionality of inheritance
- Use cases for creating subclasses of an existing class
- How to write a subclass that extends a superclass

🚀 Question of the Day

Why would I use inheritance?

💼 Key Vocabulary

- **inheritance:** an object-oriented programming principle where a subclass inherits the attributes and behaviors of a superclass
- **superclass:** a class that can be extended to create subclasses
- **subclass:** a class that extends a superclass and inherits its attributes and behaviors
- **tester class:** the class that contains the `main` method and from where the program starts running