Classes and Objects

A byte size lesson in Java programming.

Object-Oriented Language

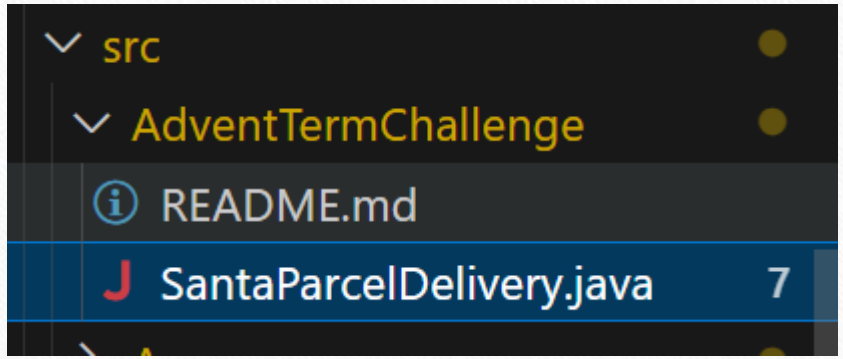
- Java is an object-oriented language that represents elements of a problem as **classes** that **contain variables** and **methods**.
- We make use of the **Keyboard class** constantly to get user input, and we also create new classes to create our programs.
- A class is simply a Java file, and now we will learn how these files can be modified and used to create **objects**.

Why do we need this?

- As our programs get more complex, we need a way to organize our code better and make it readable.

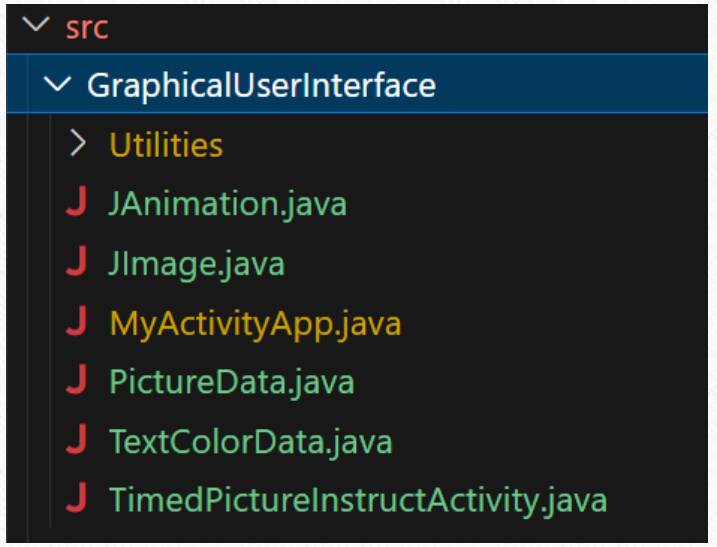
Why do we need this?

Simple Program



Single File Program

More Complex



Multiple File Program

A Simple Example

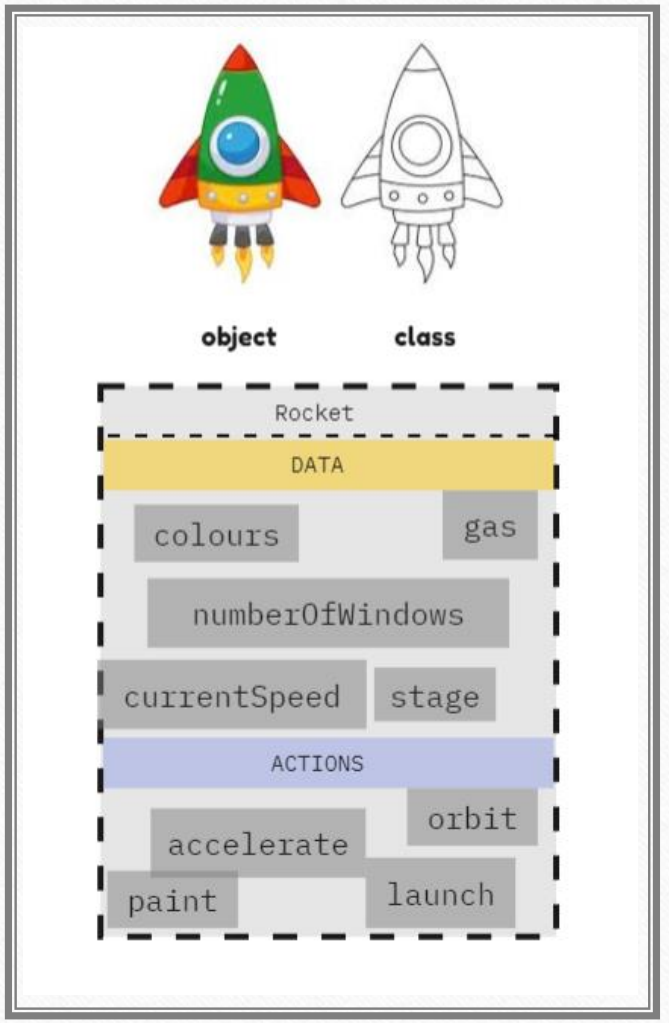
An object in Java is meant to model a real-world object.

Real-world objects have a **state** and **behaviour**.

For example, a **Rocket** has **states: gas, stage, occupancy etc**, and **behaviour: launch, accelerate, orbit, etc**.

Defining a Class

- To create **objects**, we first must **define a class**.
- A class is a blueprint, sort of like a sketch of the rocket.
- Classes contain enough detail to picture what a real-world object can look like, e.g., the number of windows.
- However, it is **just a description of what it is made of and what it can do**.



Creating Your First Class

Notice that…

- A class for an object has **data** as variables and **actions** as methods.
- There is no **main()** so it is not runnable.
- This definition only serves as a blueprint to create **instances**.

```
public class Rocket {

     // Data
    int numberOfWindows;
    double gas;

    // Actions
    public void launch() { … }

    public void orbit() { … }

}
```

Creating An Instance of Your Class

- Here is how we can create a **particular rocket** using the **new** keyword.

```
Rocket myRocket = new Rocket();
```

- From one class/template we can create as many instances as necessary.

Let's test your understanding!

- How many **classes** did we define for this program?
- From which class did we create **object instances**?

```
public class CheesecakeFactory {
    public static void main(String[] args) {
        Cheesecake newyork = new Cheesecake();
        Cheesecake chicago = new Cheesecake();
        Cheesecake philadelphia = new Cheesecake();
    }
}
```

Let's test your understanding!

- Fill in the blanks.

```
Exercise squat = new                    ();
```

```
JButton plainbutton = new               ();
```

Setting Object Data

- Different instances should describe different real-world objects. For example, we might want two rockets, one of which would be painted blue and the other would be painted red.

```
Rocket myRedRocket = new Rocket();
myRedRocket.colour = “red”;
```

object instance creation

```
public class Rocket {

     // Data
    int numberOfWindows;
    double gas;
    String colour;

     // Actions
     …
}
```

class definition

Let's test your understanding!

- Help us to **create our blue rocket** now by filling in the blanks.

```
Rocket myBlueRocket = new Rocket();
                                    ;
```

Calling Object Methods

- Objects are data structures that **group data** and **behaviour**. So naturally, after creating an instance we want to make it do something.
- We use the instance variable and call the method/action.

```
// make our rocket launch
myRocket.launch();
```

Let's test your understanding!

- Help us to **make our blue rocket orbit()**.

```
Rocket myBlueRocket = new Rocket();
myBlueRocket.colour = "red";
                                                    ;
```

Conclusion

- We have only covered the absolute basics of objects in this lesson.
- Everything we have learned before objects still applies in object-oriented programming!
- You need to solve a complex problem to combine everything you have learned so far.
- Learning about objects opens up a lot of possibilities for your final year coursework.