`class`es and Objects

Syntax

Defining a Class - *"Inventing a Composite Data Type"*

```
class <ClassName> {
    <propertyName>: <type> = <default value>;
    …
}
```

- **ClassNames** begin with an uppercase letter
- **Properties** are declared inside of the class body
  - These are *like* variable declarations without the *let* keyword
  - Properties can be assigned default values
- "A <ClassName> object will have a <propertyName> property of type <type>".
  - *"A **TwitterProfile object** will have a **followers property** of **type number**"*

Defining a Class - Example

- Here we are defining a class named **`TwitterProfile`**.

```
class TwitterProfile {
  handle: string = "";
  followers: number = 0;
  isPrivate: boolean = true;
}
```

- ***Every object*** of type `TwitterProfile` will have three properties:
  - `handle`, `followers`, and `isPrivate`

- In defining a class, you've invented a new type! You can now use it as a type. For example, in a variable declaration:

```
let aProfile: TwitterProfile;
```

Initializing a composite data type value requires **Constructing** a new object.

```
let aProfile: TwitterProfile = new TwitterProfile();
```

```
let aProfile = new TwitterProfile();
```

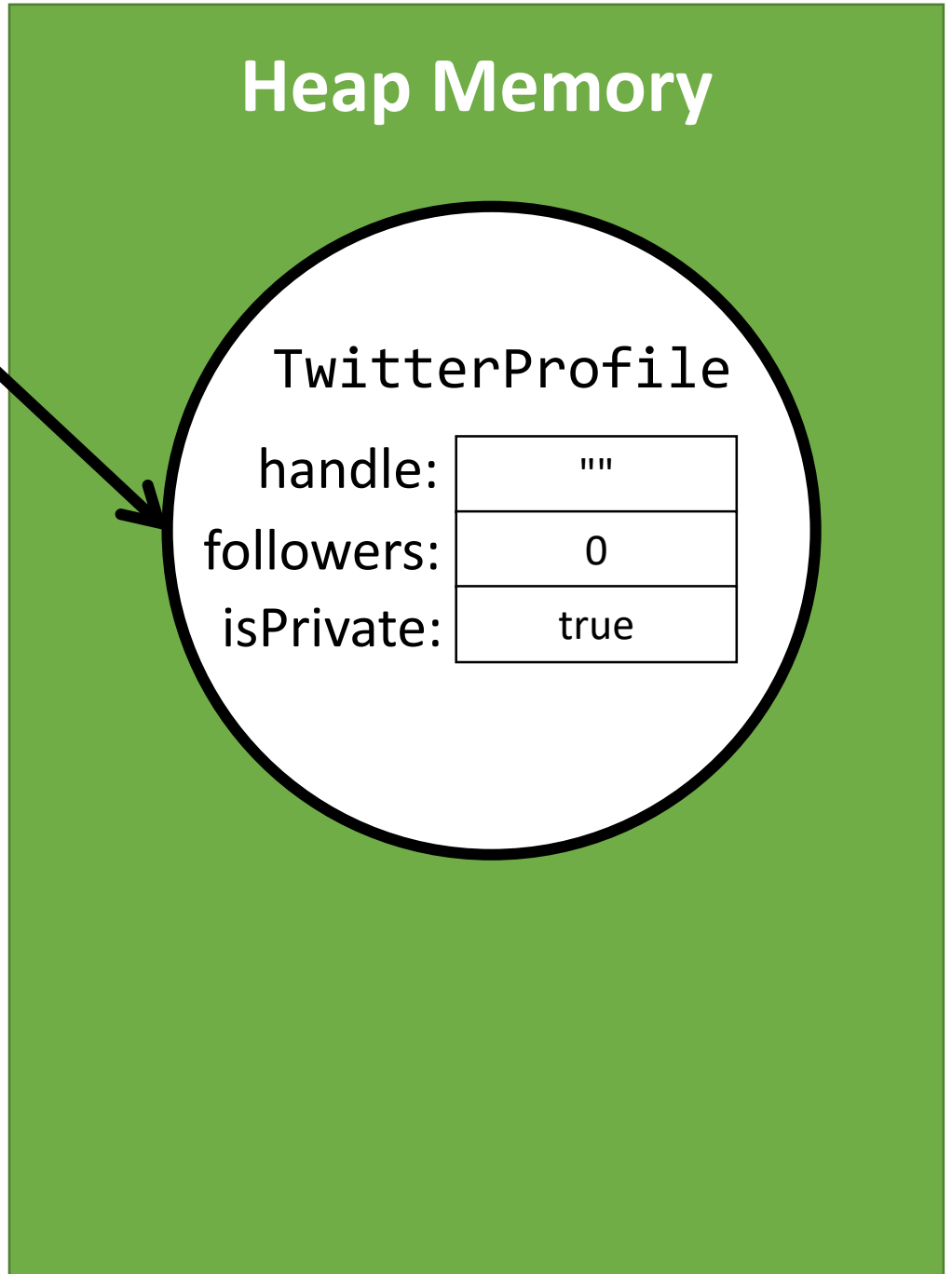
- Unlike primitives, to work with a composite data type value, a.k.a. an object, you must first **"construct"** a **new** object.

- You will write the **new** keyword followed by the class name, followed by empty parenthesis (for now).

Constructing an Object

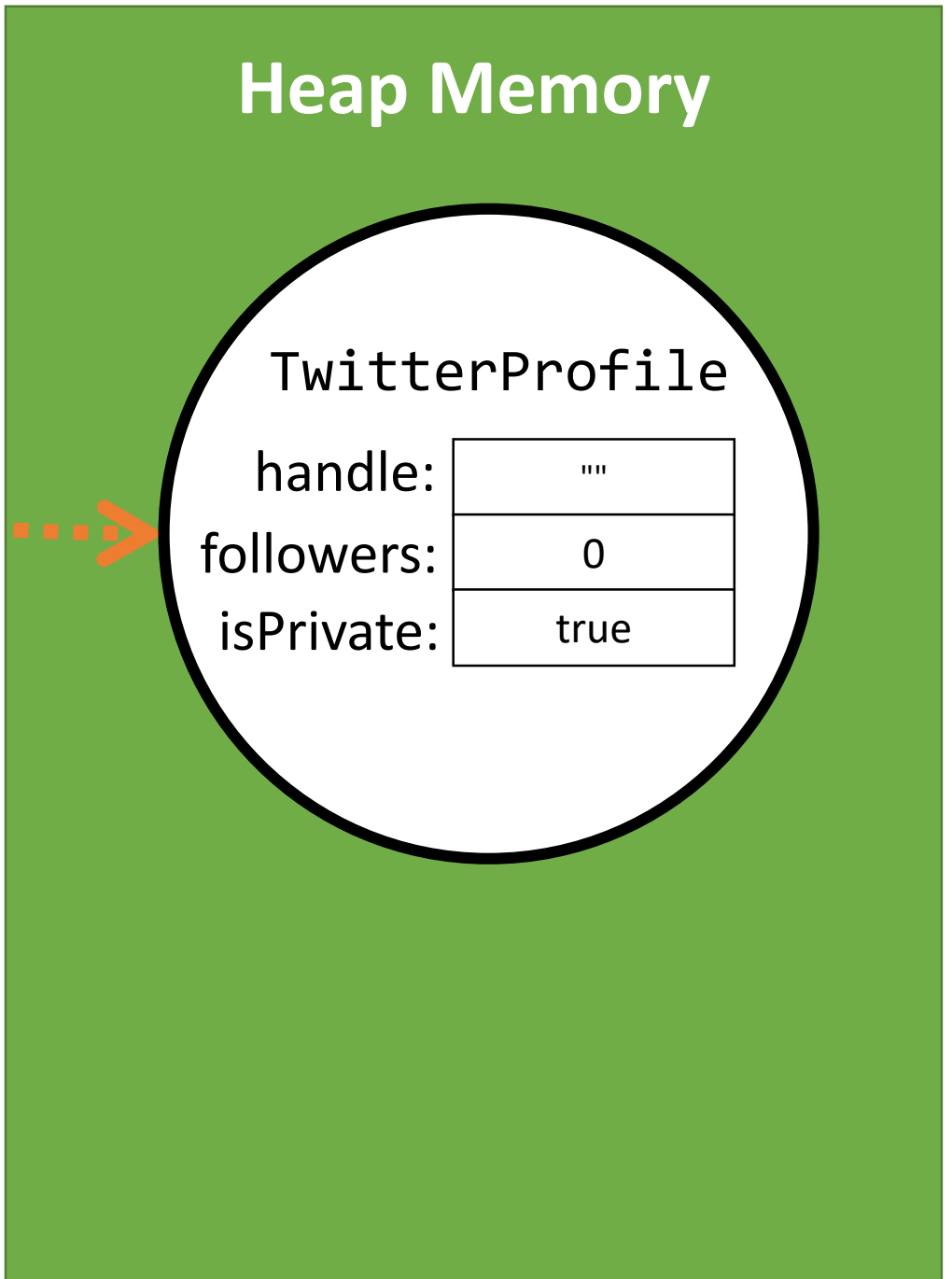
```
aProfile = new TwitterProfile();
```

- When the **new TwitterProfile()** expression is evaluated...
- ...the processor **constructs** a **new** object in heap memory with space allocated for each property.
- It also assigns the default values to each property specified in the class.
- Finally, **a reference** to this object is returned and assigned to the paper variable.
  - More on *references* soon.



Reading a Property

```
print(aProfile.handle);
```



- By referencing the TwitterProfile variable's name, followed by the *dot* operator, followed by a property name, we are saying:

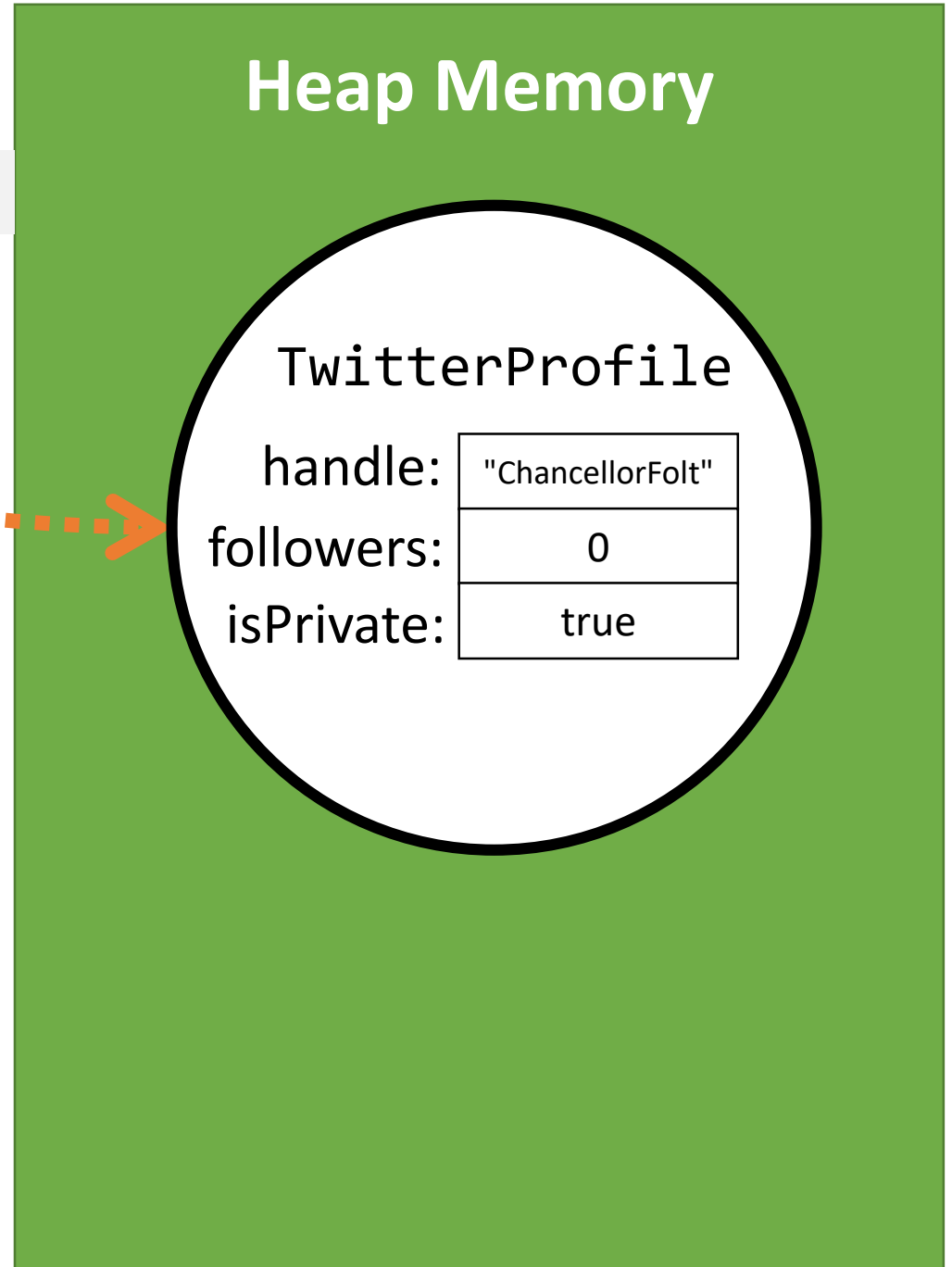
*"Hey **aProfile**, what is your **handle** property's value?"*

- General form:

  <object>.<property>

Assigning to a Property

```
aProfile.handle = "ChancellorFolt";
```



- We can change an object's property value by using the assignment operator.

*Hey **aProfile**, your **handle** is now "ChancellorFolt"*

- General form:

```
<object>.<property> = <value>;
```