Declaring variables

Instance variables

Classes can have variable declarations

One use of classes is to describe a type for objects. Objects of the same class type can have different state, and the class declares what variables are used to store such state.

```
public class Passport{

  // variable declarations

}
```

Instance variables

Variables declared in the class block are called “instance variables” (if they do not have the modifier “static”, which we’ll discuss later).

Instances are the objects created from a class, and each instance (object) can have different values in its variables, compared to other instances.

```
public class Passport{
  int height; // each Passport object has its own height value
  // other variables
  // other stuff
}
```

Declaring instance variables

As with every type of variable, the declaration contains at least two parts:

[type] [name];

For instance:

```
int height; // this passport’s height value in centimeters

String firstName; // this passport’s first name
```

Access levels

Every instance variable has an access level, declaring what the visibility of the variable is when accessed from other classes.

It is common to use the private access modifier to declare that an instance variable is only accessible (the name can only be used from) to the same class as it was declared in:

```
private int height; // only this class can use this variable
```

Public, Private

In this book, we'll only focus on two major access modifiers; public and private

Public members of a class declares names which can be used from any other class.

The public members of a class constitute the an object's API (programming interface).

The private members of a class are part of the object's implementation details which remain hidden to the rest of the world. Private parts can only be referenced by code in the class itself.

Variables are typically private

It is common to make the instance variables of a class private, which effectively lets users of the class unaware of the inner workings of the class.

We'll soon look at how variables are set (how values are assigned to variables) and how variables are used (how the values are accessible via e.g. methods).

What are the different access modifiers

`private` - only usable within the same class

`public` - usable from any class (in the same application)

If we don’t put any access modifier in front of a declaration, it is said to be “package private” or “default access”. Such names are only accessible from the same class and classes in the same package.

`protected` - we will not use this in this book, but it means “same class and subclasses and classes in the same package”

Instance variables get default values

It’s good to know that even if an instance variable never is explicitly assigned an initial value, Java makes sure they are initialized to some default value.

Numeric types get the default value 0 (or 0.0 for real types), boolean variables get the value false, and reference variables (variables whose type is for instance a class) get the value null.

What is the null value?

Reference variables are variables which can refer to objects of some type.

A reference variable which is not assigned a reference to some specific object have the special value null.

The null value means “does not refer to any object at all”.

We will later see that special care has to be taken to check whether a reference variable has a null value before trying to use the variable.

null

```
String team = "Liverpool";
```


```
String otherTeam;
```

The variable otherTeam doesn’t refer to an object, so it has the value `null`

Accessing public members

It is like we said before not common to make instance variables public, but in order to show you how to access an instance variable from a different class, we'll make a small class with a public instance variable:

```
public class Member{
  public String name; // a public instance variable
}
```

Next, we'll show you how to make a Member object and access the name variable (note that this only works because we made name public!)

Creating a Member and setting name

```
public class TestMember{
  public static void main(String[] args){
    Member m = new Member();
    m.name = "Adam"; // to access a public instance member
                     // use the reference and a dot followed
                     // by the name of the public instance
                     // member (in this case a variable
                     // called name)
  }
}
```

But you said private is more common?

Yes, instance variables usually are private. But making them private would make it illegal to access the variables from the test class.

So, if we can’t access the variables, how do they get intialized?

The next lecture will show you how, using so called “constructors”.