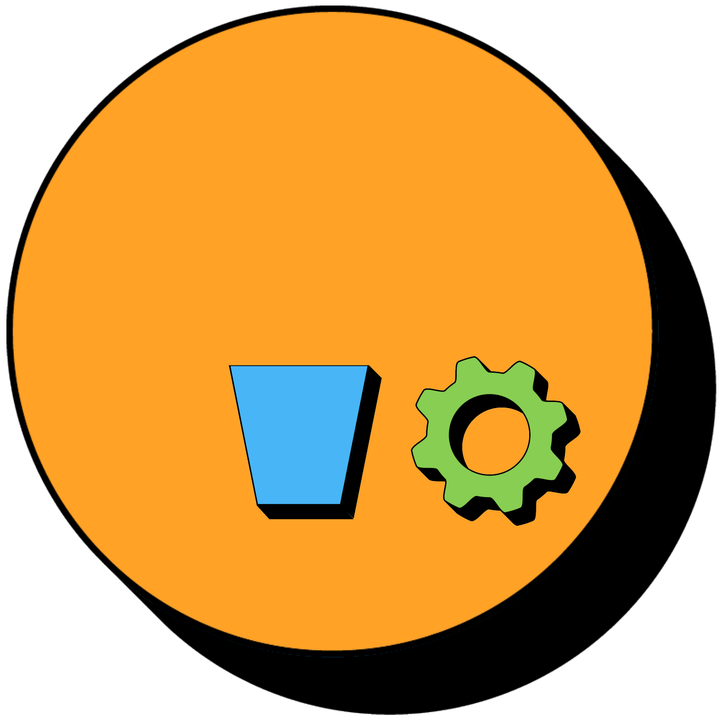
Essential Computing 1

Information hiding



Public, **protected** and **private** access modifiers

- **Public** variables and methods are accessible from everywhere.
- **Protected** variables and methods are accessible within the same package (default).
- **Private** variables and methods are accessible only within the same class.

Without an access modifier, Java falls back to the **default: protected**.

```
// This will only work if the code is
// placed in same package, or no package
// is defined.
Person p = new Person();
System.out.print( e.birthYear );
```

```
public class Person {
    int birthYear;

    Person(){
        birthYear = getCurrentYear();
    }
}
```

Protected

In the case of "birthYear", it would make sense if it could not change (be immutable).

```
Person p = new Person();

System.out.print( e.birthYear );

// Should not be allowed!
p.birthYear = 2008;
```

```
public class Person {
    int birthYear;

    public Person(){
        birthYear = getCurrentYear();
    }
}
```

Making "birthYear" private and crating a public **getter method**

```
Person p = new Person();

System.out.print( p.getBirthYear() );
```

```
public class Person {
    private int birthYear;

    public int getBirthYear(){
        return birthYear;
    }

    public Person(){
        birthYear = getCurrentYear();
    }
}
```

Making a **setter method** that allows overwriting "birthYear" if the value is sane.

```
Person p = new Person();

System.out.print( p.getBirthYear() );

// Using a safe setter method.
p.setBirthYear( 2008 );
```

```
public class Person {
    private int birthYear;

    public int getBirthYear(){
        return birthYear;
    }

    public void setBirthYear( int year ){
        if( year < 1900 ) return;
        if( year > getCurrentYear() ) return;
        birthYear = year;
    }

    {...}
}
```