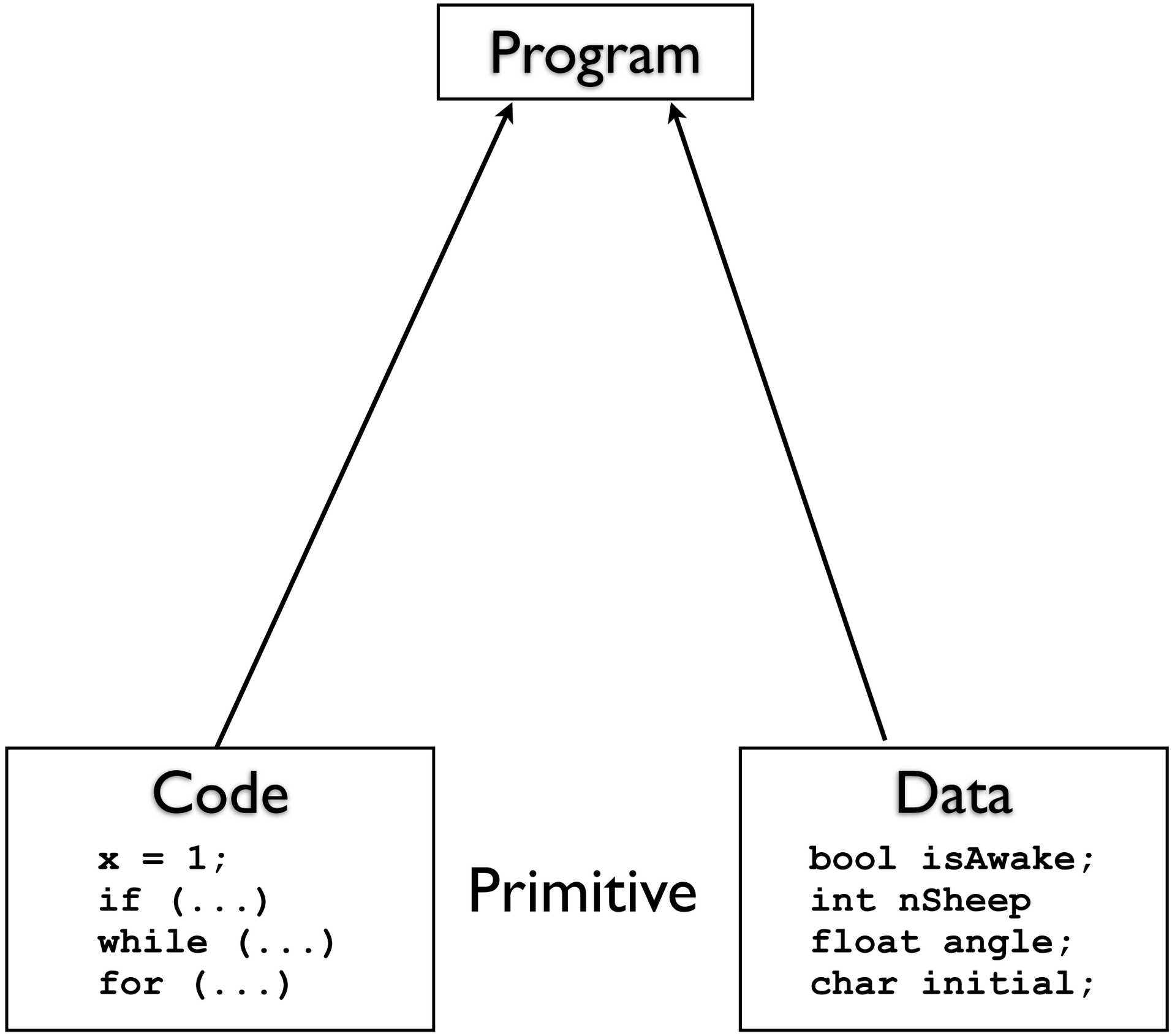
COMP1400

Week 7 - Object oriented programming


Program
Code
x = 1;
if (...)
while (...)
for (...)
Primitive
Data
bool isAwake;
int nSheep
float angle;
char initial;

Program

Compound

Methods

```
public int sort(
      int[] data)
```

Arrays

```
int[][] pixels;
char[] name;
```

Code

```
x = 1;
if (...)
while (...)
for (...)
```

Primitive

Data

```
bool isAwake;
int nSheep
float angle;
char initial;
```

Abstraction

Compound code and data structures provide us a means of abstraction - i.e dividing the program into chunks.

Each chunk contains a collection of related code and/or data. We give the chunk its own name and identity.

Abstraction

Abstraction has several advantages:

- Code re-use
- Clearer design
- Easier debugging

Blocks of duplicated code or data is usually a sign of a need for abstraction.

Example -
One big method

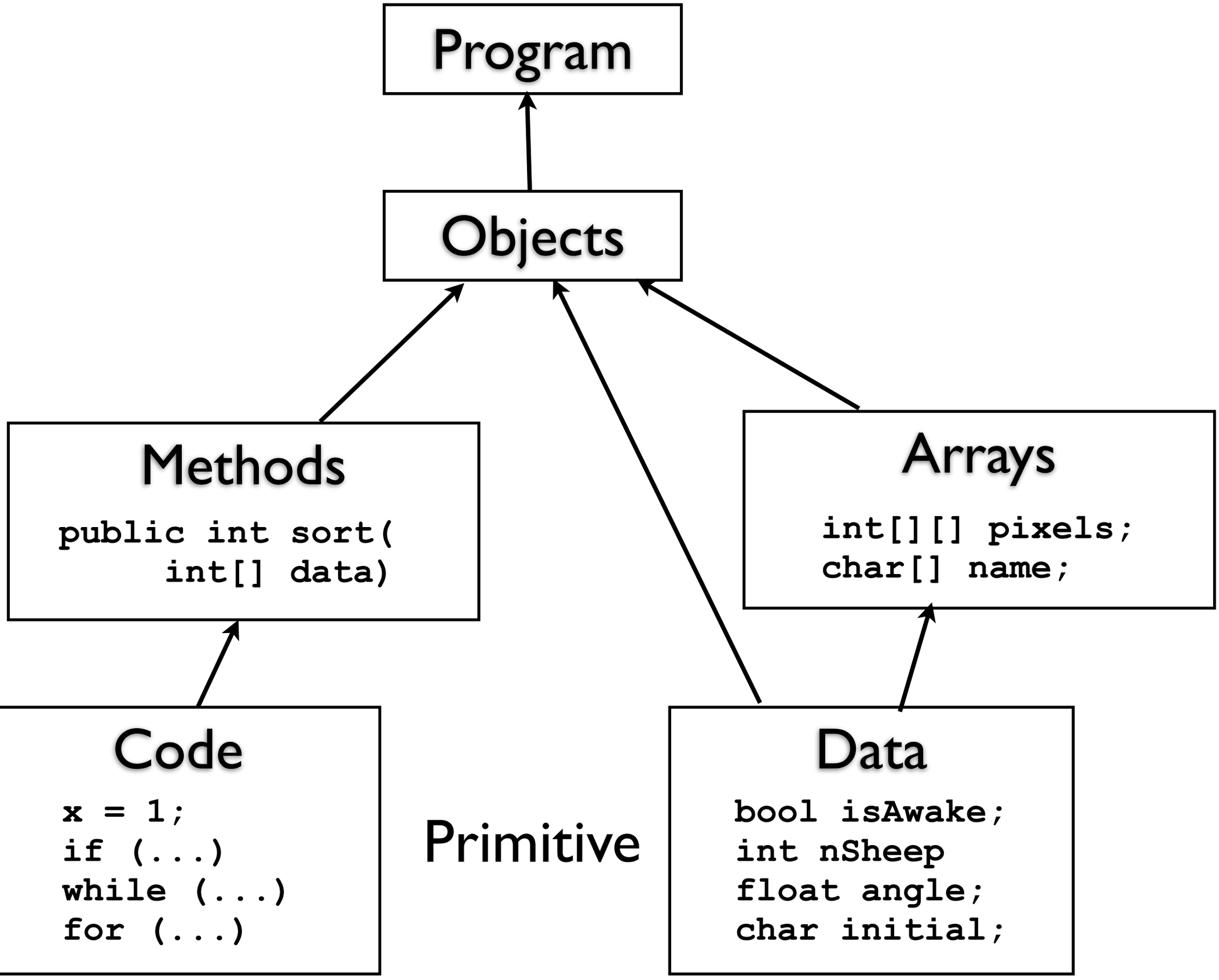
Objects

An object is a abstract chunk of code and data that belong together.

Example: A variable size list of Strings

Data	Code

`String[] contents;`
`int size;`
	`void add(String item)`
	`void remove(String item)`
	`String get(int pos)`


Program
Objects
Methods
public int sort(
int[] data)
Arrays
int[][] pixels;
char[] name;
Code
x = 1;
if (...)
while (...)
for (...)
Primitive
Data
bool isAwake;
int nSheep
float angle;
char initial;

Object-oriented Programming

Object-oriented programming (OOP) is the process of designing and implementing programs as systems of interacting objects.

It is fundamentally based around the ideas of abstraction and encapsulation.

Object-oriented Programming

Abstraction

- Dividing the program into chunks at different levels of detail.

Encapsulation

- Each chunk hides its implementation details from outsiders.

Program

Object

```
// primitive
bool isAwake;
int nSheep;
```

Object

Object

```
// primitive
bool isAwake;
int nSheep;
```

Object

```
// primitive
bool isAwake;
int nSheep;
```

Object

```
// primitive
bool isAwake;
int nSheep;
```

Object

```
// primitive
bool isAwake;
int nSheep;
```

Program

Object

hidden

Object

hidden

Object

hidden

Encapsulation

Each object has:

- a public interface that describes how it can be used
- a private implementation that describes how it works

Classes

Objects have types called Classes.

Classes represent all objects of a certain kind.

All objects in a class share a common structure but have different details.

Example

"Car" is a class.

All cars have data:

- colour
- engine capacity, etc.

and methods:

- drive
- park, etc.

Example

My car is an object.
It is an instance of the class Car.

My car has data with specific values:

colour = yellow

engine capacity = 1.5 litre

and methods:

drive

park, etc.

Static data and methods

You usually need a specific instance of a class (an object) to access data and methods.

Eg: you can't drive the class "Car" or ask what colour "Car" is.

However some methods and data belong to all cars and can be run on the class. These are called "static".

Java Class Library

Java comes with a large collection of classes for common object types.

You can browse the library online at:

http://docs.oracle.com/javase/6/docs/api/index.html?overview-summary.html

Random

The Random class is one

ArrayList

Another important class is ArrayList.

http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html

This class represents variable length lists. Internally it is implemented using arrays but it provides a much more flexible interface.