classes + objects

- recall each object has
  - state: info remembered by an object
  - behaviour: abilities of an object
- ex: coin
  - state: outcomeVisibility, isFake
  - behaviour: greyOutImage

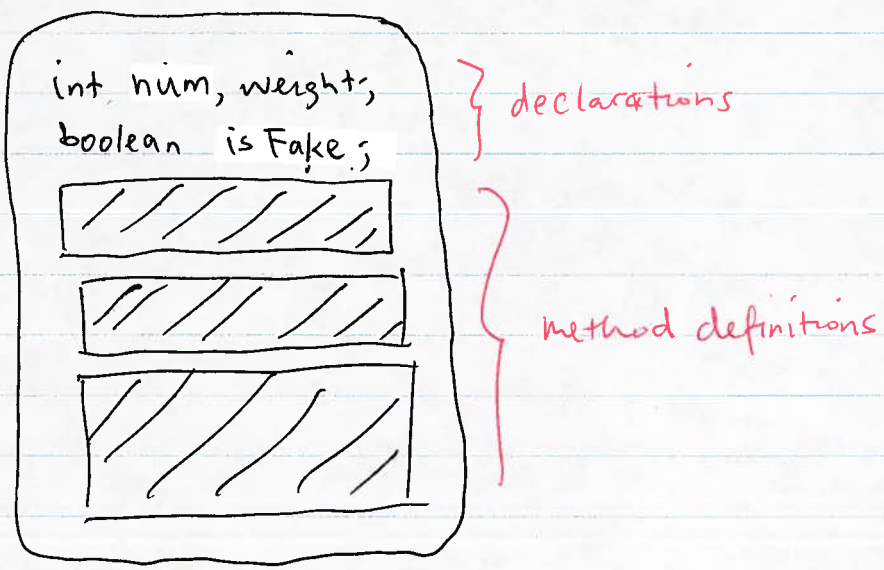
- game can create one or more coin objects
- nt: each object's attributes can remember diff values

- group of similar objects characterized by a class definition

- class skeleton review:

```
class className
{
    // attributes
    // methods
}
```

- visually:



Special method: constructor

- used to setup an object upon creation
  - initialize attribute values
  - may do other setup work

- method name must be same as class name

ex:
```
class Game
{
    // attributes
    Game() {...}
    // other methods
}
```

- good practice to always include a constructor in your class even if nothing to initialize or setup ★

Special method: toString()

- template to use:

```
String toString()
{
    String str = "";
    // gather object info into one phrase
    return str;
}
```

- returns object info as a String
- helps reveals object state
- good practice for debugging purposes ★

More About Methods

- declare a method with a method header or signature
- ex:

```
char calc (int num1, int num2, String message)
```

char → return type; calc → method name; (int num1, int num2, String message) → parameter list

- header is followed by method body where you define the process/routin
- ex:

```
char calc ( int num1, int num2, String message)
{
    int sum = num1 + num2;
    char result = message.charAt ( sum);
    return result;
}
```

local data → (more later)

return statement

- all methods with something to return must have a return statement
  - statement must return something of the same type as method's declared return type

- what type is used when a method has nothing to return?
  - void
- exception: constructor
  - blank return type: don't write anything
- why do constructors have nothing to return?

Parameters

- when method is called, we invoke it with actual parameters with values to be used inside the method

caution: simplified

```
class TestDemo
{
        Demo example = new Demo();
        :
        int count = 5;
        char letter;
        letter = example.calc( 2 , count, "hello there");
}
```

- actual parameters are copied into the formal parameters in method definition

```
class Demo
{
        :
        char calc(int num1, int num2, String message)
        {
                int sum = num1 + num2;
                char result = message.charAt(sum);
                return result;
        }
}
```