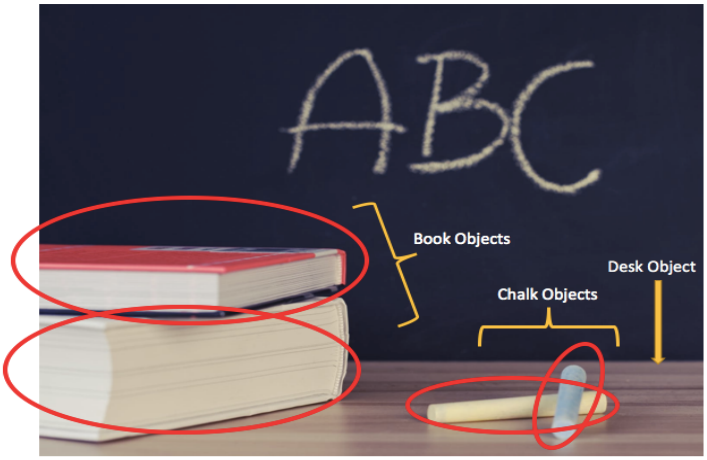
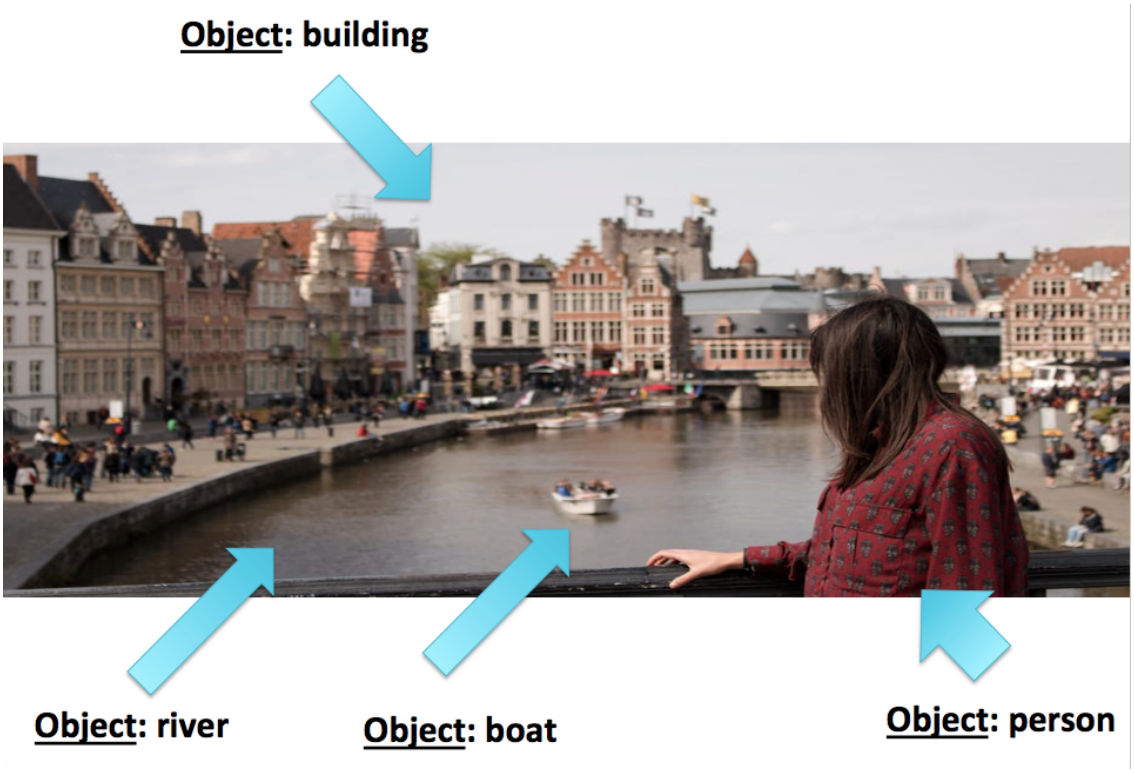
Classes/Encapsulation (+Stroustrup's glossary)

http://www.stroustrup.com/glossary.html#Gencapsulation

Object-oriented Paradigm:

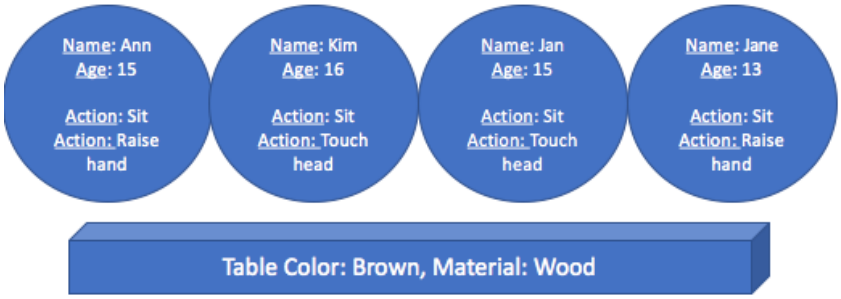
- Before we start, let's consider the idea of the object-oriented paradigm
  - Simply put, we consider things around us as objects:





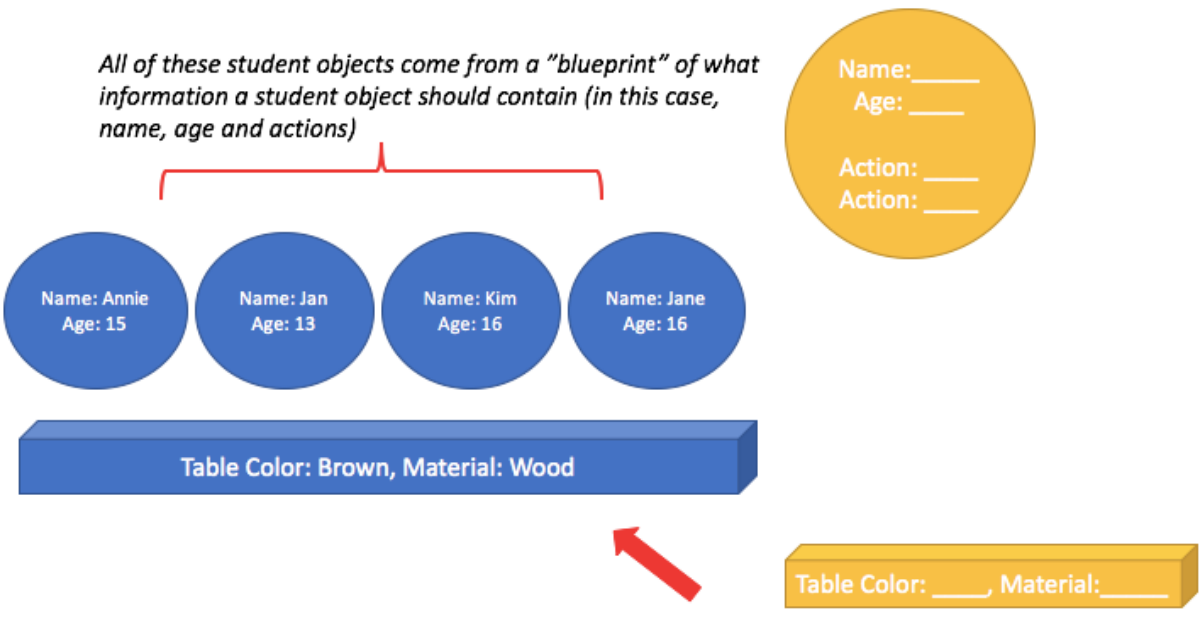
- Just like in real life, an object can have characteristics (state of an object), perform actions and have actions performed on it (behavior of an object). For example:

Real world: 4 students sitting on a table

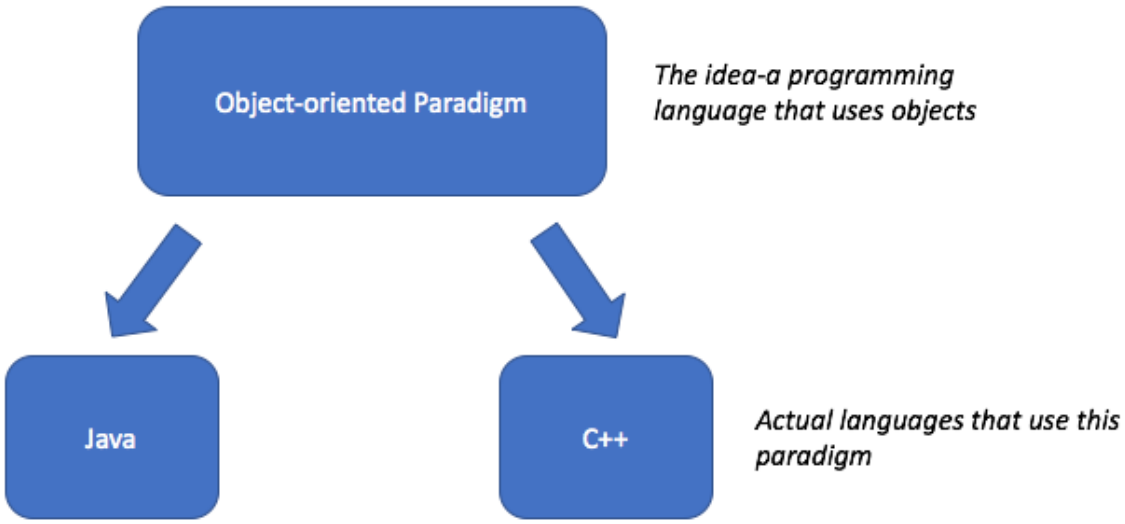


Real world represented as objects (this is what we turn into code)

- When we create an object, we use a "blueprint" or "template" (called a class in C++) to create an object:



- Notice that this paradigm is used as a basis for different programming languages
    - We could be teaching this class about object-oriented programming in another language that uses the idea (this class used to be taught in Java)



- C++ gives us the **option** to use the object-oriented paradigm (C did not give us this option built into the language)
    - Notice that you could actually build a fully functioning program in C++ without creating objects

---

Classes & Encapsulation:

- A class in C++ is the "blueprint" mentioned above
    - We create the objects we are dealing with using the blueprint
    - We will talk about this in more detail next class

- Encapsulation is the idea of bundling up (encapsulating):
    - The state of an object (characteristics at any given time-name and age in our student example above) AND
    - Behavior of our object (actions like raising hand in our student example above)
    - We can also limit outside access to states and behavior to the outside word (information hiding)
    - Check glossary given above for Stroustrup's definition

- In C++, the encapsulation idea can be implemented by creating a class
    - This class contains the state of the object in the form of member variables
    - This class contains the behavior of the object in the form of member functions

- C was not made with this idea in mind
  - We could code in an object-oriented fashion in C, but features are built into C++ for this purpose

Sample Programs:

- Sample Program 1
  - ABC Treasure Company searches for treasure chests in the ocean. Treasure chests always contain pearls, diamonds and coins. Create a program for them to compare two treasure chests to see which is worth more.
    - What information do we need? How much each pearl, diamond and coin is worth (take user input)
    - For simplicity in class, I will hardcode total found into the program (but know you could take it as input)

    - Thinking with objects: what objects do we see?
      - Remember different people may decide to use different objects based on how they view the problem
      - I will only make a treasure chest objects
        - Note that I could have made pearl, diamond and coin objects, but I feel this problem does not need that (it is enough for me to use variables in the treasure chest). In a more advanced problem, I might have done so.

Sample Run:

```
computer$ g++ treasure.cpp
computer$ ./a.out

~Treasure Calculator~
Enter the price of each pearl:  $2.45
Enter the price of each diamond:  $3.90
Enter the price of each coin:  $1.89

-Total in treasure chest 1:  $42.77
-Total in treasure chest 2:  $ 27.61

First treasure chest has more.
```

Code:

```
#include <iostream>
#include <iomanip> //used for setprecision() below
```

```
using namespace std;

class Treasure_chest{

  public:
    int num_pearls;
    int num_diamonds;
    int num_coins;

  //total value a treasure chest has
    float find_total(float p, float d, float cn)
    {
        return (num_pearls*p)+(num_diamonds*d)+(num_coins*cn);
    }
};

int main(int argc, char ** argv)
{
     cout << "\n~Treasure Calculator~\n";
     float price_pearls, price_diamonds, price_coins;
     cout << "Enter the price of each pearl:  $";
     cin >> price_pearls ;

     cout << "Enter the price of each diamond:  $";
     cin >> price_diamonds ;

     cout << "Enter the price of each coin:  $";
     cin >> price_coins ;


     Treasure_chest first_found;
     Treasure_chest second_found;

     //we give the total numbers of pearls, diamonds and coins for each treasure chest
     first_found.num_pearls=4;
     first_found.num_diamonds=7;
     first_found.num_coins=3;

     second_found.num_pearls=5;
     second_found.num_diamonds=2;
     second_found.num_coins=4;

     float total_first=first_found.find_total(price_pearls, price_diamonds, price_coins);
     float total_sec=second_found.find_total(price_pearls, price_diamonds, price_coins);

     cout << "\n-Total in treasure chest 1:  $" << fixed << setprecision(2) <<total_first<<endl;
     cout << "-Total in treasure chest 2:  $ " << total_sec<<endl;

     if(total_first<total_sec) //sec worth more
     {
        cout << "\nSecond treasure chest has more." << endl;
```

```
    }

    else if(total_sec<total_first) //first worth more
    {
       cout << "\nFirst treasure chest has more." << endl;
    }

    else //worth the same
    {
       cout << "\nBoth are worth the same amount." << endl;
    }
}
```

- Sample Program 2
  - ABC Animal Shelter has asked you to create a program that can input new found pets into the system and display all pets in the system to potential families for adoption.
    - What information is important? Name, breed, age, vaccinations

---

    - Thinking with objects: what objects do we see?
      - I see puppy objects so I will make a Puppy class
        - A more advanced problem might see a shelter object that keeps puppy objects inside

Sample Run:

```
Programs computer$ g++ -o pup puppy.cpp
Programs computer$ ./pup
***ABC Animal Shelter***

found or see?
see
No animals yet!


found or see?
found
Name: Spike
Breed: German Shepherd
Age in Weeks: 8
Vaccinate? yes
Puppy added!


found or see?
found
```

```
Name: Rex
Breed: Lab
Age in Weeks: 12
Vaccinate? no
Puppy added!


found or see?
see
**All puppies available:**

--Spike!--
-Breed:German Shepherd
-Age in Weeks:8
-Puppy had shots.

--Rex!--
-Breed:Lab
-Age in Weeks:12
-Puppy has not had shots.

found or see?
exit
Exiting...
```

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

//puppy class
class Puppy{

    string name;
    string breed;
    int age_weeks;

  public:
    bool shots; //has the puppy had all its shots?

    void give_info();
    void print_info();
};

//functions are defined outside of the class
void Puppy::give_info() {

  string answer;

  cout << "Name: ";
  getline(cin,name);

  cout << "Breed: ";
  getline(cin,breed);
```

```
  cout << "Age in Weeks: ";
  cin >> age_weeks;

}

void Puppy::print_info() {

   cout << "--" << name << "!--\n";

   cout << "-Breed:" << breed << "\n";

   cout << "-Age in Weeks:" << age_weeks << "\n";

}


//this function prints out all puppies available (just show you could do this-ideally all functions should be
in a class)
void print_puppies(vector <Puppy> animal_shelter)
{
   cout << "**All puppies available:**\n" <<endl;

   for(int i = 0; i < animal_shelter.size(); i++)
   {
   animal_shelter[i].print_info();

   if(animal_shelter[i].shots)
   {
   cout << "-Puppy had shots.\n\n";
   }

   else
   {
   cout << "-Puppy has not had shots.\n\n";
   }

   }
}


int main(int argc, char ** argv)
{
    vector <Puppy> animal_shelter;
    string answer="yes";

    cout << "***ABC Animal Shelter***\n" << endl;

    //options: found a dog, get a dog
    while (answer.compare("exit")!=0)
    {
       cout << "found or see?" << endl;
```

```
    getline(cin,answer);

     //puppy is found-add to system
    if(answer.compare("found")==0)
    {
         Puppy p;
         p.give_info();
         getchar();
         cout << "Vaccinate? ";
         getline(cin,answer);

          if(answer.compare("yes")==0)
          {
                p.shots=true;
          }

         else
         {
                 p.shots=false;
         }

         animal_shelter.push_back(p);

         cout << "Puppy added!\n\n" << endl;
     }

    //see all puppies available
    else if(answer.compare("see")==0)
    {
         if(animal_shelter.size()==0)
         {
            cout << "No animals yet!\n\n" << endl;
         }

         else
         {
                 print_puppies(animal_shelter);
         }
    }

 //exit
else if (answer.compare("exit")==0)
{
          cout << "Exiting..." << endl;
}


else
{
         cout << "Not a valid input.  Enter found or see.\n\n" << endl;
 }
```

```
  }

}
```