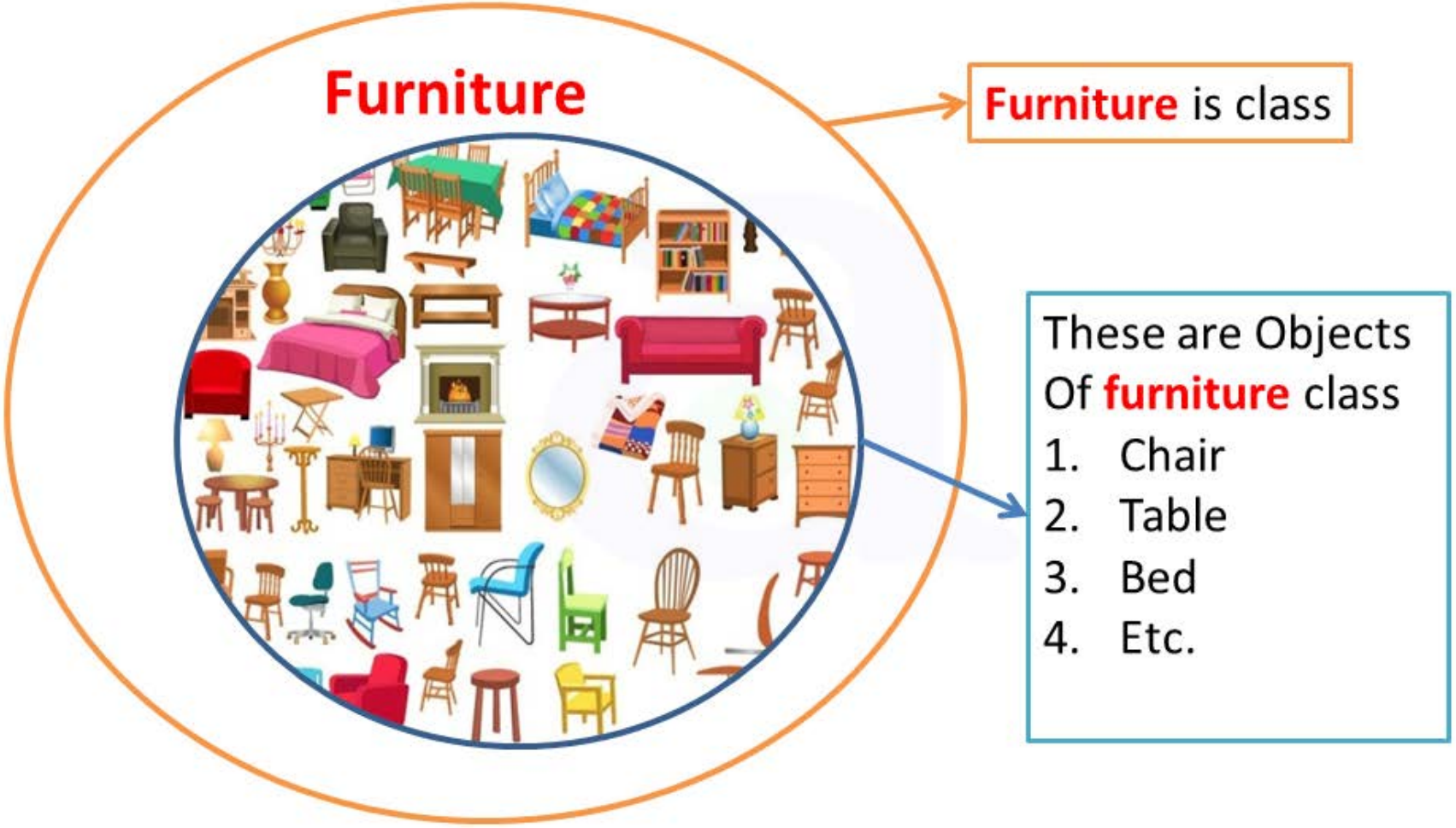
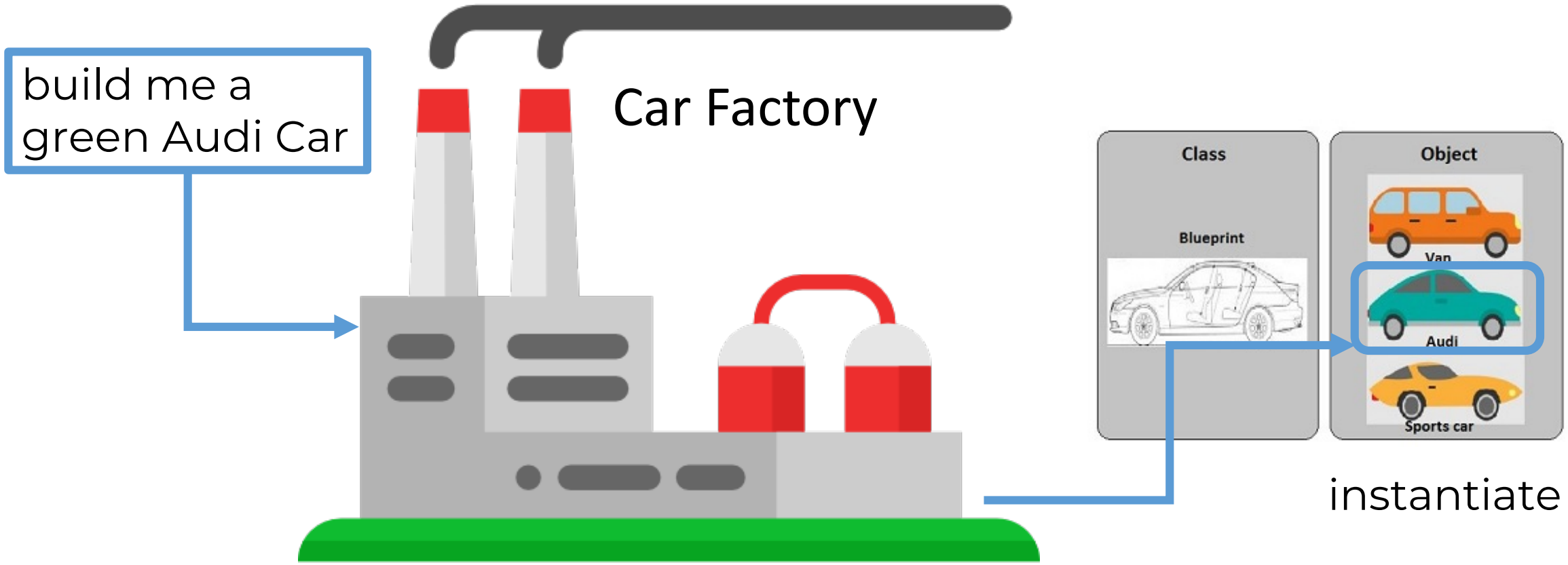
objects and classes

DAY 8

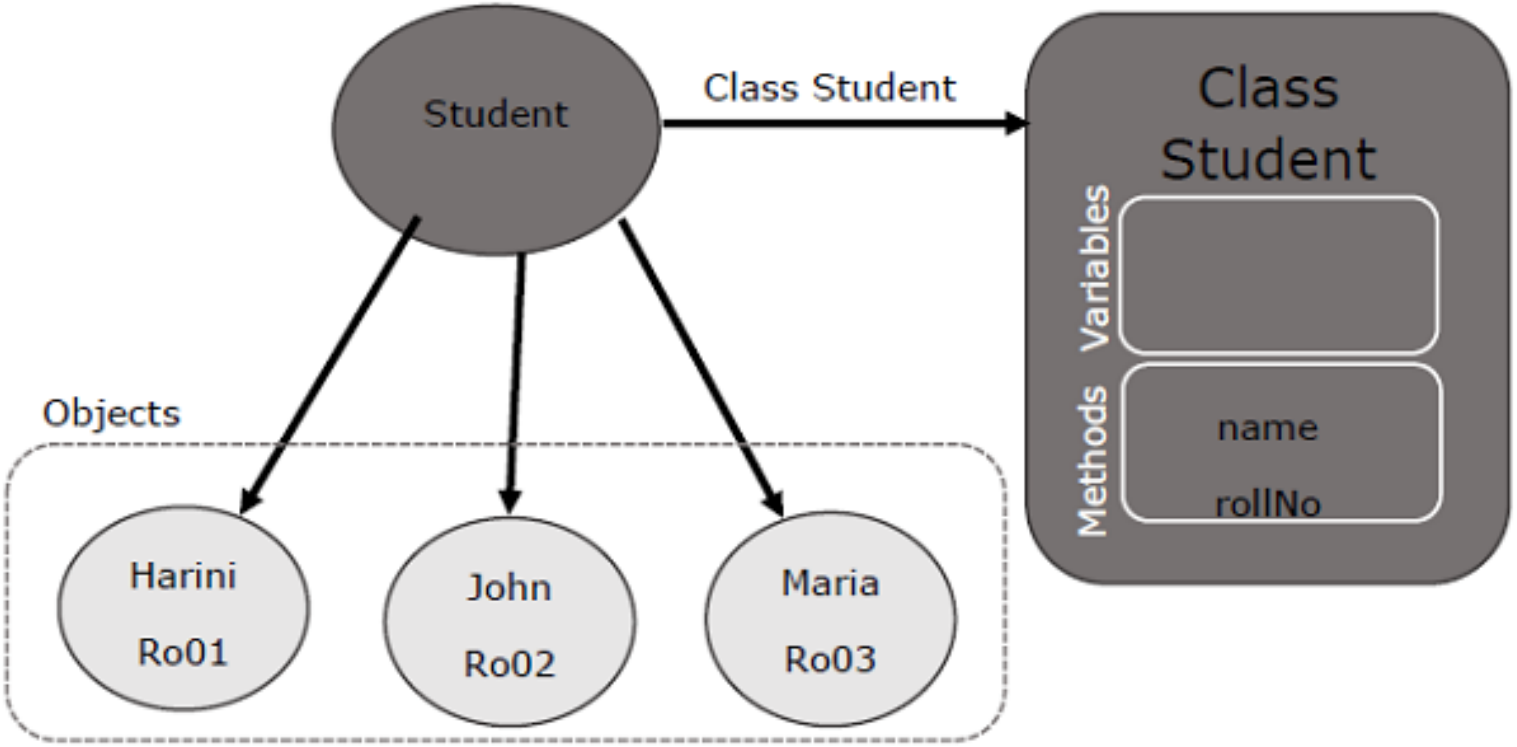
classes are templates.
objects are specific instances of a class.

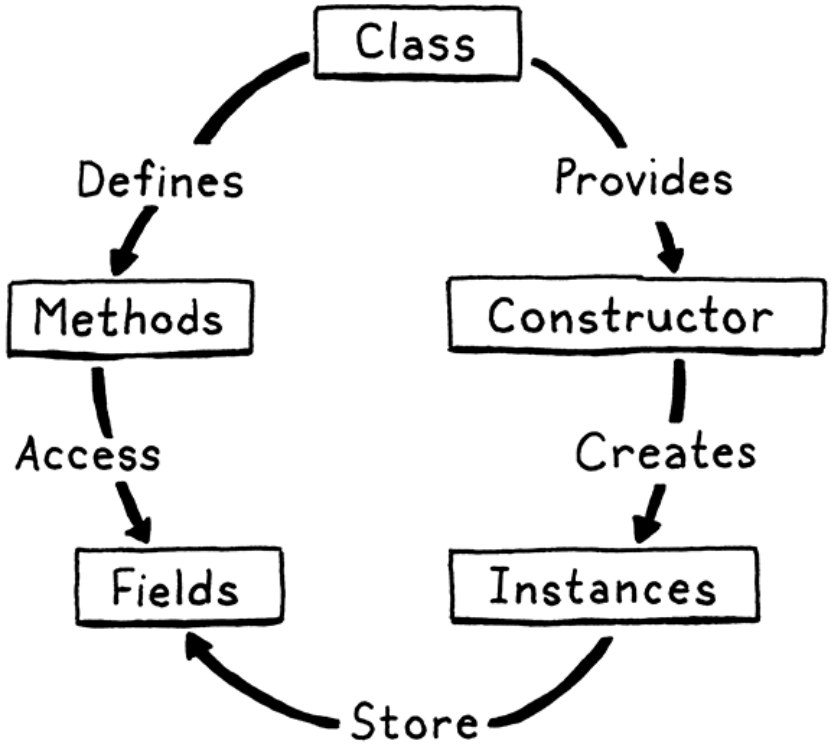


classes - factories with instructions on how to construct an object when invoked by program.

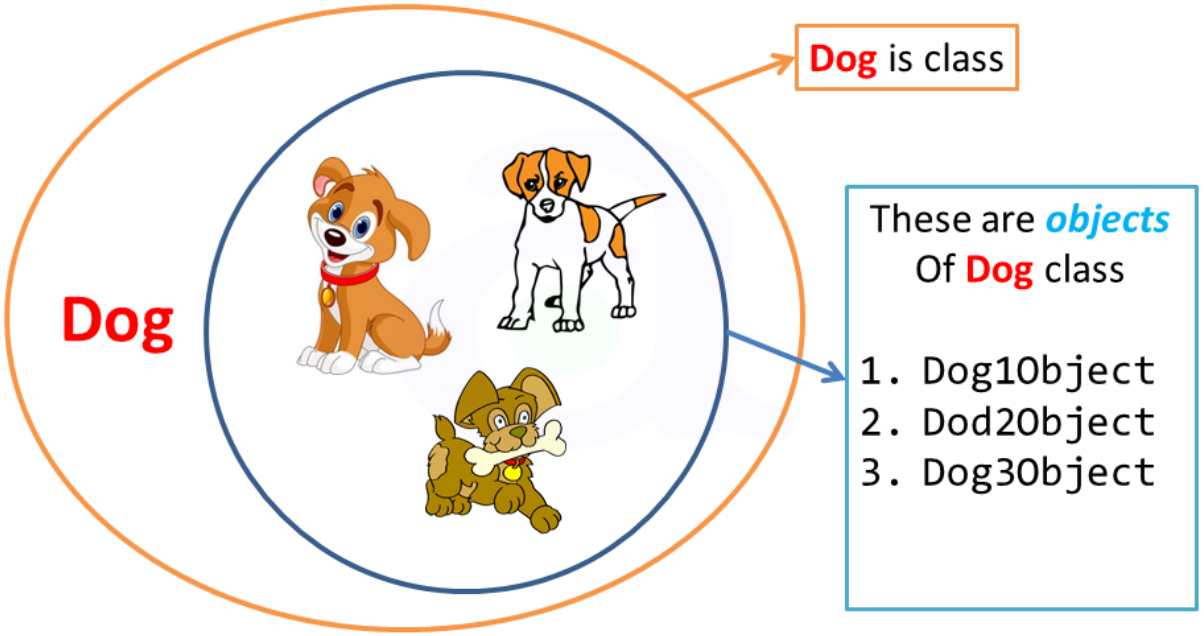


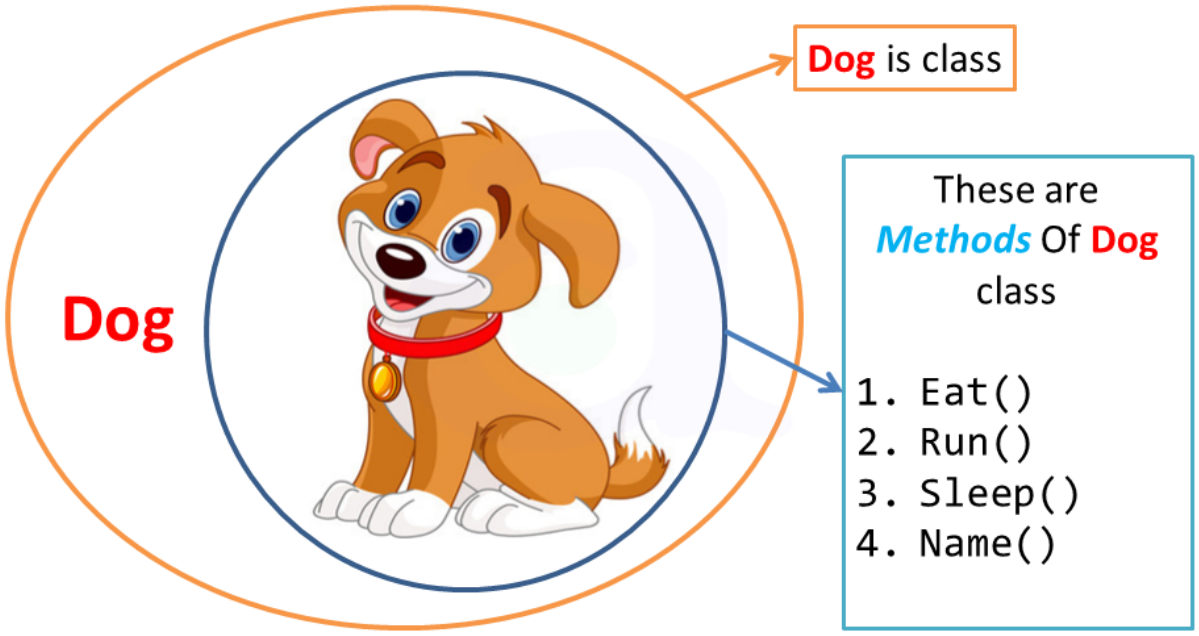
objects - a specific member of the class; made of variables, functions (methods), constructor.





objects are specific instances of a class. methods are actions an object can perform.





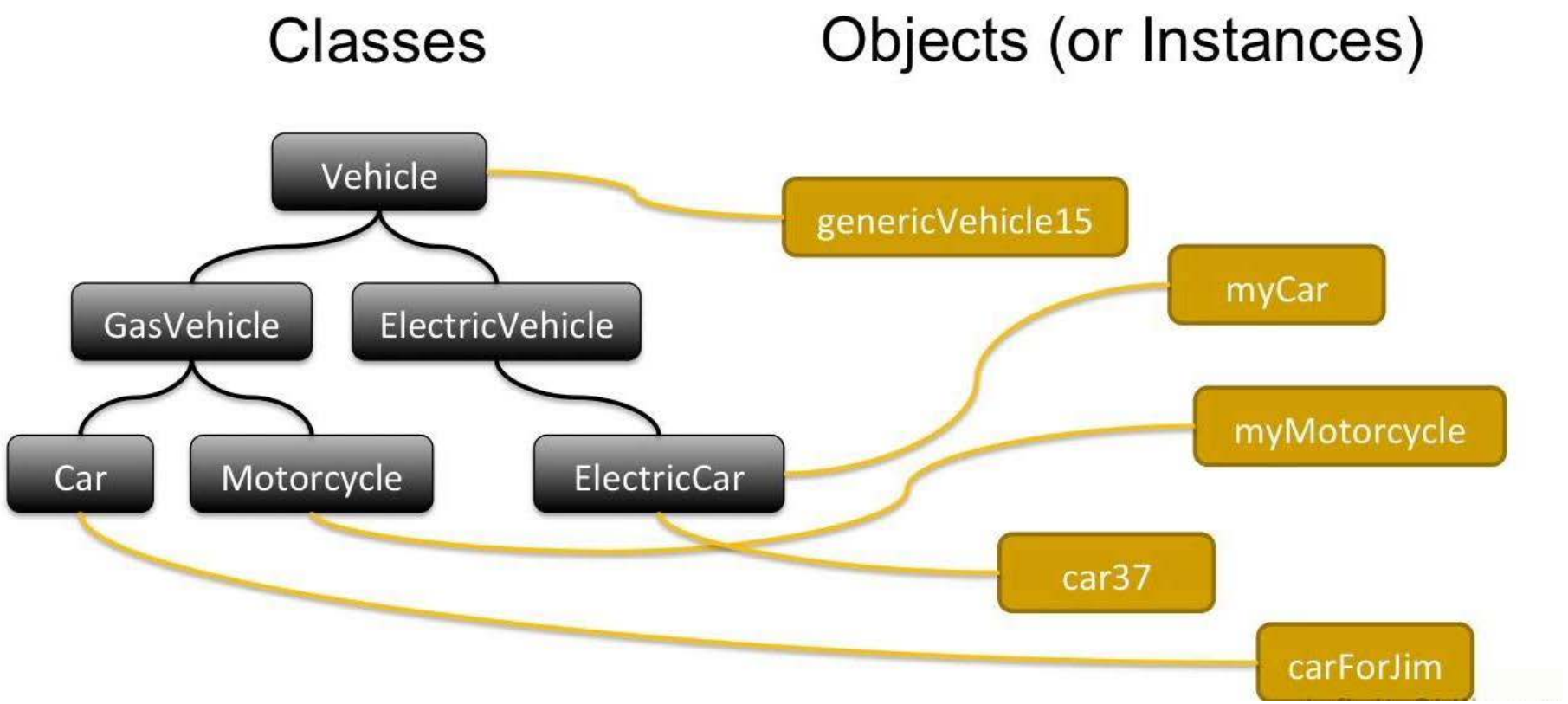
Plan

- 1. specify the class - methods - variables,
- 2. call the class to instantiate objects.

- But first, group exercise!

Exercise

What are some examples of classes, objects, and methods in everyday (or not so everyday) life?



Methods (functions)?

```
StartEngine();

StopEngine();

Open(int door);

Close(int door);

Turn(int direct);

etc...
```

Bouncing Ball: object oriented version

Main_BouncingBall_Program | BouncingBall_Class

```
//Main BouncingBall Program

void setup() {
  size (600,600);
  smooth();
}

void draw () {
  background (0);
}
```

Main_BouncingBall_Program | BouncingBall_Class

```
//NAME OF CLASS
class BouncingBall {



  //VARIBLES



  //THE CONSTRUCTOR



 //FUNCTIONS

}
```

myBall, an instance of class BouncingBall

Main_BouncingBall_Program | BouncingBall_Class

```
class BouncingBall {

  //VARAIBLES (Note: these variables are global to the class, you are just declaring them here)
 float x = 0;
 float y = 0;


 //THE CONSTRUCTOR (Note: the constructor runs only once when called it is not a looping function)
 BouncingBall(){
 }


 //FUNCTIONS
 void display () {
   ellipse (200,200,20,20);
 }


}
```

Main_BouncingBall_Program | BouncingBall_Class

```
//Declared
BouncingBall myBall;<-- DECLARE A CLASS LIKE A VARIABLE
                    In previous lessons, to
                    declare variables, we
                    preficed the names of
                    our variables with the type.
                    Often we used "Int" or "Float"
                    and then followed the type
                    with a name of our choice.
                    Now, to declare OBJECT
                    CLASSES, the class type is
                    the actual name of the class
                    (in this case BouncingBall)
                    and then you follow the type
                    with a name of our choice.
                    I use "myBall" to declare my
                    class below.
//Initializes
void setup() {
  size (600,600);
  smooth();
  myBall = new BouncingBall ();   <-- The keyword "new" is saying
}                                     build a new instance of the class.


//Functionality
 void draw () {
  background (0);
  myBall.display(); <--This is a function of the class, the dot syntax
                        opens up the class and its functions. In this case
}                       we have one function, display()
```

Main_BouncingBall_Program | BouncingBall_Class

```java
class BouncingBall {

  //GLOBAL VARIABLES (Specific only to the BouncingBall class)
 float x = 0;
 float y = 0;



 //THE CONSTRUCTOR
 BouncingBall(float _x, float _y){    // <-- Here, we are adding the variables _x and _y inside the constructor.
                                     // _x and _y are only visable to the constructor. We need _x and_y
                                    //  as place holders so we can manipulate our objects in the main program.


   x = _x;   // <-- We also want our objects to access the functionality below. To do this
   y = _y;  //      we set our _x and _y place holders to equal our class's global variables x and y.
 }


 //FUNCTIONS
 void display () {
   ellipse (x,y,20,20);  // <-- We replace the first two values in the ellipse with x and y so we can have
                        //      choices for its fuctionality.
 }


}
```

Main_BouncingBall_Program | BouncingBall_Class

```java
//Main BouncingBall Program


//Declared
BouncingBall myBall;



//Initializes
void setup() {
  size (600,600);
  smooth();
  myBall = new BouncingBall (400,400); // <-- The constructor has two place marker arguments, _x and _y.
                                      //      So, we must add in two arguments here to match the number of place
                                     //       markers. Here we add in 400,400 to change the position of our ellipse.
 }



//Functionality
 void draw () {
  background (0);
  myBall.display();
}
```

Main_BouncingBall_Program | BouncingBall_Class

```java
class BouncingBall {

  //GLOBAL VARIABLES (two additions)
 float x = 0;
 float y = 0;
 float speedX = 4;
 float speedY = .5;

 //THE CONSTRUCTOR (no change)
 BouncingBall(float _x, float _y){
   x = _x;
   y = _y;
 }

  //FUNCTIONS (3 more functions added)

 void move() { //move the ball around the screen
  x = x + speedX;
  y = y + speedY;
}

 void bounce(){ //bounce the ball when you get to the edges.
  if ((x > width) || (x < 0)){
  speedX = speedX * -1;
  }
  if ((y > height) || (y < 0)){
    speedY = speedY * -1;
  }
  }

 void gravity(){ //mimic gravity by changing speed on y axis.
 speedY = speedY + 0.2;
 }

  void display () { //display the ball
    ellipse (x,y,20,20);
  }
 }
```

Main_BouncingBall_Program | BouncingBall_Class

```
//Main BouncingBall Program


//Declared
BouncingBall myBall;




//Initializes
void setup() {
  size (600,600);
  smooth();
  myBall = new BouncingBall (400,400);
 }




//Functionality
 void draw () {
  background (0);
   myBall.display();
  myBall.move();
 myBall.bounce();
myBall.gravity();
}
```

Run

Main_BouncingBall_Program | BouncingBall_Class

```
class BouncingBall {

   //GLOBAL VARIABLES
  float x = 0;
  float y = 0;
  float speedX = 4;
  float speedY = .5;

  //THE CONSTRUCTOR
  BouncingBall(float _x, float _y){
    x = _x;
    y = _y;
  }

   //FUNCTIONS

   void run(){ // New comprehensive function called run that lumps them all together in one move.
     move();
     bounce();
     gravity();
     display();
   }

 void move() {
  x = x + speedX;
  y = y + speedY;
}

 void bounce(){
  if ((x > width) || (x < 0)){
  speedX = speedX * -1;
  }
  if ((y > height) || (y < 0)){
    speedY = speedY * -1;
  }
  }

 void gravity(){
 speedY = speedY + 0.2;
 }
  void display () { //display the ball
    ellipse (x,y,20,20);
  }
 }
```

Main_BouncingBall_Program | BouncingBall_Class

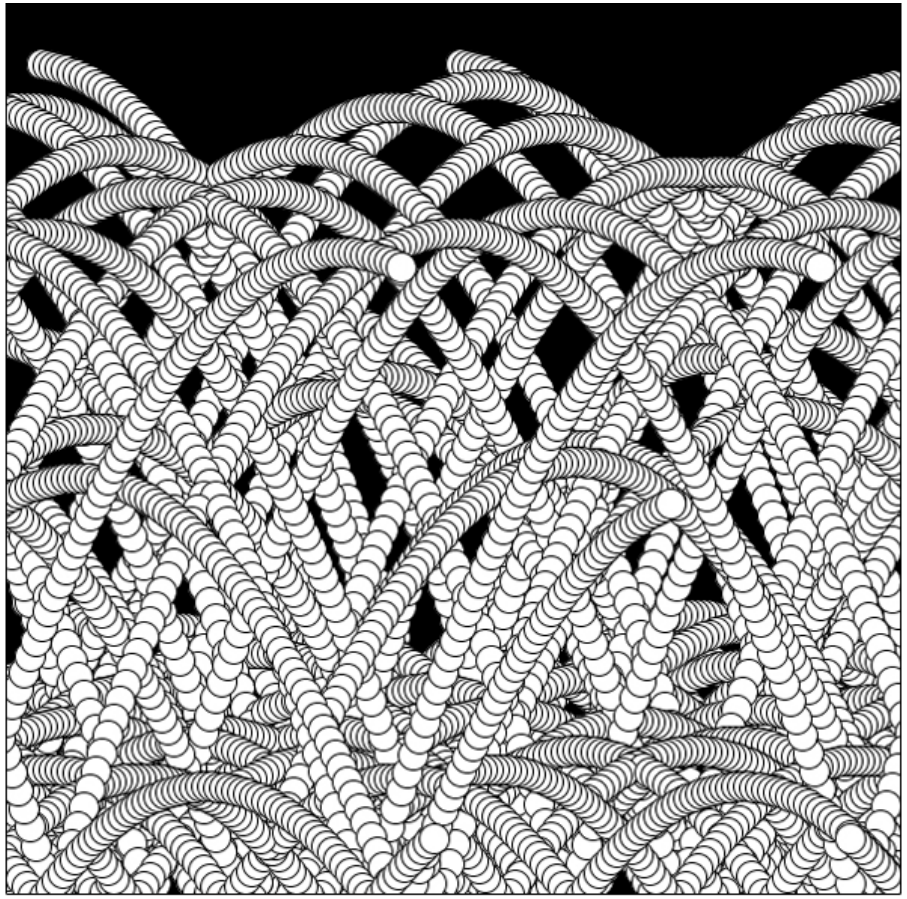
```
//Main BouncingBall Program


//Declared
BouncingBall myBall;




//Initializes
void setup() {
  size (600,600);
  smooth();
  myBall = new BouncingBall (400,400);
 }




//Functionality
 void draw () {
  background (0);
   myBall.run();  //Just one function needed here now, run()
}
```

Main_BouncingBall_Program | BouncingBall_Class

```
//Main BouncingBall Program


//Declared
BouncingBall myBall;
BouncingBall myBall1;
BouncingBall myBall2;
BouncingBall myBall3;
BouncingBall myBall4;


//Initializes
void setup() {
  size (600,600);
  smooth();
  myBall = new BouncingBall (400,400);
  myBall1 = new BouncingBall (30,40);
  myBall2 = new BouncingBall (100,200);
  myBall3 = new BouncingBall (10,400);
  myBall4 = new BouncingBall (300,300);
 }




//Functionality
 void draw () {
  background (0);
   myBall.run();
 myBall1.run();
myBall2.run();
myBall3.run();
myBall4.run();
}
```

Exercises (for fun)

- Make an array of BouncingBalls and put them at regular spacing around the screen.
- Show the balls' movements (trails) as they are influenced by the physics.



objects and classes

DAY 8