|  |  |
| --- | --- |
| **Date Assigned: 10/21/15** | **Date Due: 10/23/15** |
| **Unit:** Methodology | **Turn In List:** **1. Terms 2. Cars pde** |
| *“I will harness the power of object oriented design in programming with a simple example of a car.”* | |

**Classes and Objects**

**Content Objectives:** Students will begin to explore classes and objects in Processing.

|  |
| --- |
| **Starter Activity** |
| This entire assignment is based on the Processing tutorial on objects: <http://processing.org/tutorials/objects/> -  Code the example from the car class demo from lecture on a 2nd tab in a new project. |

|  |  |
| --- | --- |
| **Key Terms:** | |
| OOP | Object-oriented programming (OOP) languages, such as C++ and Java, provide a formal set of rules for creating and managing objects. |
| Class | a class is a template definition of the method s and variable s in a particular kind of object . Thus, an object is a specific instance of a class; it contains real values instead of variables. |
| Objects | a class is a template definition of the method s and variable s in a particular kind of object . |
| Properties | property, in some object-oriented programming languages, is a special sort of class member, intermediate between a field (or data member) and a method. |
| Constructor | A constructor is a special method of a class or structure in object-oriented programming that initializes an object of that type. |
| Functions | a task (they combine many instructions into a single line of code). Most programming languages provide many built in functions that would otherwise require many steps to accomplish, for example computing the square root of a number. In general, we don't care how a function does what it does, only that it "does it"! |

|  |
| --- |
| **Assignment:** |
| Create a new “Tab” in a Processing project and name it Car (yes with a capital C.) Write the Car class as described in the Objects article under “Writing the Cookie Cutter” with comments naming the four code blocks. Once your car class is created, add 3-5 elements to the look of the car (visible tires, body parts etc…) Your car is now a Data Type of its own!  Challenge #1: the example code at the conclusion of the article has two cars (see : <http://processing.org/tutorials/objects/>)  Challenge #2: can you use random strategically to assign car colors, speed, or even size?  Challenge #3: can you make the cars go in both directions on the screen?  Challenge #4: Can you use an array of the Car object and create 100 of them?  Challenge #5: Add two or more details to the appearance of the car (i.e. windows, tires, antenna etc…)  Paste code below for a real world class you created with the following:   1. Name 2. Member variables 3. Constructor 4. Visibility Method 5. Behavior Method |

Notes (Points of interest, mistakes, lessons learned, web resources, and thoughts):

|  |
| --- |
| Car [] myCars= new Car[40];  Car [] myCars= new Car[40];  Car myCar1;  Car myCar2;  void setup() {  size(1920,1080);  myCar1= new Car(128,random(width),random(height),random(50));  for(int i=0; i<myCars. length;i++ ){  myCars[i]=new Car(128,random(width),random(height),random(50));    }  }  void draw() {  background(255);  myCar1.display();  myCar1.drive();  for(int i=0; i<myCars. length;i++ ){  myCars[i].display();  myCars[i].drive();  }  class Car {  color c = color(0);  float x ;  float y ;  float speed ;  float left;  Car(color tempC, float tempX, float tempY, float tempSpeed) {  c = tempC;  x = tempX;  y = tempY;  speed = tempSpeed;  left=random(2);  }  void display() {  fill(c);  rect(x, y, 30, 10);  }  void drive() {  if (left <=1) {  x+= speed;  if (x>width) {  x=0;  }  } else {  x -= speed;  if (x<0) {  x=width;  }  }  }  } |