Custom Objects and Methods

Object vs. Instance

* What is the difference between the two?

|  |  |
| --- | --- |
| **Difference between an Object and an Instance** | |
| Object (Human) | Instance (Lupoli) |
|  |  |

Introducing the Theory of Objects

* PROGRAMMER DEFINED data types
* int, double, double, etc… all hold ONE variable’s value

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lupoli\_name |  | Lupoli\_depart. |  | Lupoli\_title |  | Lupoli\_salary |
| Mr. Lupoli |  | Comp. Sci. |  | Assist. Prof. |  | -1 |

String Lupoli\_name = “Mr. Lupoli”;

String Lupoli\_department = “Computer Science”;

String Lupoli\_title = “Assistant Professor”;

int Lupoli\_salary = -1;

* we would have to create a separate variable for each, even though the variables are ALL related!!

|  |
| --- |
| EMPLOYEE |
| name |
| department |
| title |
| salary |

* Objects
  + groups related variables into ONE table/object
  + creating your OWN data type/template
  + need to only create one TEMPLATE for each person or “instance”

Why use Objects?

# Java is based in Objects!!!

* + calls them classes (reserved word)

# basic data types

* good data type to JUST hold data, very basic

Where have you seen the reserved word “class” before?

Syntax and setup

|  |
| --- |
| **Class Code (Employee.java)** |
| public class Employee Employee   |  | | --- | | name | | department | | title | | salary |   {  public String name;  public String department;  public String title; data members  public int salary;  }  // Employee literally becomes a data type like int, char, etc… |
| **Class setup** |
| AFTER creating your class for the “main”, add ***another*** class of whatever object you need.    Inside Employee.java, notice:   1. no main() 2. inside same project (2 files within the SAME project) |

In a new project, create the files “Driver.java” and “Employee.java”. (Capitalization matters!!) “Driver” has the main. Copy the code above for Employee.

|  |  |
| --- | --- |
| **Eclipse Project Setup** | |
| Driver.java | Employee.java |
| **public** **class** Driver {  **public** **static** **void** main(String[] args)  {    }  } | **public** **class** Employee  {  **public String** name;  **public String** department;  **public String** title;  **public int salary;**    } |

Creating an instance in the main

* ALL instances are created in the main/DRIVER!!!!!
* Creating an instance is when you create a variable of your new type

|  |
| --- |
| **First Example of an instance** |
| import // whatever  **public** **class** Driver {  **public** **static** **void** main(String[] args) {  Employee EM001 = **new** Employee();  EM001.name = "Mr. Lupoli";  EM001.department = "Computer Science";  EM001.title = "Assistant Professor";  EM001.salary = -1;  Employee EM002 = **new** Employee();  EM002.name = "Steve Thomas";  EM002.department = "Computer Science";  EM002.title = "Web Designer";  EM002.salary = 120000;  }  } |

|  |
| --- |
| **The overall idea – The Big Picture** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | EMPLOYEE TEMPLATE |  | EM002 |  | EM001 |  | EM003 | | name |  | Mr. Hughes |  | Mr. L |  | Jenny | | department |  | C.S. |  | C.S. |  | Admin. | | title |  | TA |  | Assist. Prof. |  | Site Dir. | | salary |  | -100 |  | -1 |  | 100000000 |   Each of these are an instance of our Employee data type that we created!! |

|  |  |
| --- | --- |
| **Eclipse Project Setup** | |
| Driver.java | Employee.java |
| **public** **class** Driver {  **public** **static** **void** main(String[] args)  {    Employee Lupoli = **new** Employee();  // just created a new instance  Lupoli.name = "Mr. Lupoli";    // create your own instance!!!      System.***out***.println(EM002);  System.***out***.println(EM002.name);  }  } | **public** **class** Employee  {  **public String** name;  **public String** department;  **public String** title;  **public int salary;**    // what if no public/private in front??  }  USE PUBLIC FOR NOW |

1. Create 3 ***instances*** using the Employee class, of yourself, your parents, or a friends’ info (can be fake info) in JAVA: ***(DO NOT RECREATE THE CLASS!!)***
2. Run the project
3. **Draw** which each would look like:
4. See if you can display the values in your instance. See if you can figure it out.
5. Run the project

Compiling and naming classes

* There should be ***AT LEAST*** two files now in your projects
  + main/driver
  + class/methods
  + please look BELOW for naming scheme
* remember the name of the file should match the name of the class

|  |  |
| --- | --- |
| **Example Files and Setup** | |
| Class File | Driver File |
| **Employee.java** | **Driver.java** |
| #import // whatever  class Employee  {  member variables;  …  member methods;  } | #import // whatever  class Driver  {  public static void main(…)  {  Employee EM003 = new Employee();  …  }  } |

ALWAYS COMPILE FROM THE MAIN/DRIVER!!! NO EXCEPTIONS!!

|  |
| --- |
| Class 3  Class 1  (file)  Driver/Main()  Class 2  (file) |

Custom Methods within Objects

* just like helper methods, custom methods can be created for custom Objects
* the code for methods are added below the data members in the class code
* again like helper methods, custom methods require
  + return value
  + name
  + parameters (if necessary)
    - remember, what the method needs in order to work
* methods are also broken down into mutators and accessors
  + mutators CHANGE **the value** data members
  + accessors just RETRIEVE (no change) data members

|  |
| --- |
| **Object/Class with Methods** |
| **public** **class** Employee  {  **public String** name;  **public String** department;  **public String** title;  **public int salary;**    // what if no public/private in front??  **public String** getName( ) { **return** name; }  **public String** getDepartment( ) { **return** department; }  **public String** getTitle( ) { **return** title; }  **public** **int** getSalary( ) { **return** salary; }  **public** **void** setSalary( **int** newSalary) { salary = newSalary; }  **public String** toString( )  { **return** name + "\n" + department + "\n" + title + "\n" + salary;}  } |

|  |
| --- |
| **Complete Main example using the Class Employee** |
| **import** java.util.Scanner;  **public** **class** Driver {  **static** Scanner *sc* = **new** Scanner(System.*in*);    **public** **static** **void** main(String[] args) {    Employee Heila = **new** Employee(); // just created a new instance  Heila.name = "Ms. Heila";  Heila.department = "Garbage duty";  Heila.salary = -11111;  Heila.title = "dumpster cleaner";    // hard code way of setting her salary  Heila.setSalary(10);  // user input way of setting her salary (Scenario #1)  System.*out*.println("Please place in Heila's new salary");  **int** nS = *sc*.nextInt();  Heila.setSalary(nS);  // user input way of setting her salary (Scenario #2)  System.*out*.println("Please place in Heila's new salary");  Heila.setSalary(*sc*.nextInt());  }  }   1. In Employee, add and create ALL mutators and accessors!!! (Use the templates above)    1. ***No toString function yet!!*** 2. In the main(), create another instance of your TA!!! I want you to get user input for your TA’s values. 3. Display those values at the end of the main(). 4. WHY DOESN’T NAME, DEPARTMENT, and SALARY need to be declared, OR an INSTANCE in front WITHIN the functions?? 5. Identify a parameter in ANY of the mutator functions. 6. Identify the return type of the method getSalary()? 7. Identify the following about the above object’s methods  |  |  |  |  | | --- | --- | --- | --- | | name of method | return type | parameter (if any) | what does it do? | | getName() | String |  |  | |  |  |  |  | |  |  |  |  | |

What is overloading??

* methods that share the exact method name, but different parameters, and possibility different NUMBER of parameters

|  |
| --- |
| Overloaded Example |
| **public** **class** Example {  **public** **void** getMethod(**int** score) // Method #1  {  System.***out***.println( "Method #1" );  System.***out***.println( score );  }  **public** **void** getMethod (**char** grade) // Method #2  {  System.***out***.println( "Method #2" );  System.***out***.println( grade );  }  **public** **void** getMethod (**double** average) // Method #3  {  System.***out***.println( "Method #3" );  System.***out***.println( average );  }  **public** **static** **void** main(String [] args )  {  Example ex = **new** Example();    ex.getMethod(98); // What method # will be called??  ex.getMethod(12.3); // What method # will be called??  ex.getMethod('F'); // What method # will be called??  }  } |

The ToString Function

* overloads the String function “toString”
* created by the programmer
* used to display the instance and all of it’s values
* function is added to the class
* NOTICE no “”toString()” behind the instance!!
  + called automatically when System.out.println(String) is called

|  |
| --- |
| **ToString Function Example** |
| Code |
| **public** String toString()  {  **return** getfirstName() + ", " + getlastName() + "\n " + getAge();  } |
| Called |
| **public** **class** Driver  {  **public** **static** **void** main(String[] args)  {  Employee adjunct = **new** Employee("Shawn", "Lupoli", 21);  Employee dean = **new** Employee("Jack", "McLaughlin", 75);  Employee professor = **new** Employee("Super", "Mario", 81);    System.*out*.println(adjunct);  }  } |
| Output |
| Shawn Lupoli  21 |

Go ahead and add the “toString” method to Employee. Have it return their name and salary only. Within the main(), have ONE of your instances printed.

Access modifiers (public and private)

* encapsulation is the concept of determining what items of data should be EASILY accessible or HARD to get to
  + access from the main/driver
* there are two types (actually more, but later)
* public
  + **methods** and **variables** that ***CAN*** be accessed by the class AND main()
* private
  + **methods** and **variables** that ***CAN NOT*** be access by the main, but CAN BE from INSIDE the class
* Strategy, what would ***YOU*** want people to be able to see?
* It is safer to be private

|  |  |
| --- | --- |
| Bank Account | Automobile |
| |  | | --- | | Long int/String account\_number | | Double interest rate | | Double balance | | String account\_type | | Long int routing\_number | | Int/String SSN | | **DATE** Open date | | String bank\_name | | Boolean current | |  | |  | |  | | |  | | --- | | Mileage (int) | | Vin# (String) | | Model (String) | | Make (String) | | Year (int) | | Brake\_type (String) ABS, standard… | | Drive\_tran (Boolean) automatic,.. | | Plate # (String) | | Horsepower (int) | | Max\_speed (String/Int) | | NumOfWindows | | NumOfDoors | |

Look at all data members, as a group, decide on which are public or private

Notice: From here out all data members will be PRIVATE in my notes

Encapsulation – Class setup

* We are using a method to change the values, NOT changing them directly!!
* Need a method to return the actual value

|  |
| --- |
| **What the class code should look like** |
| **class** Employee  {  **private** String name;  **private** String department;  **private** String title;  **private** **int** salary;  // mutators  **public** **void** setName(String newName) { name = newName; } // set name  **public** **void** setDepartment(String newDP){department = newDP;}// sets department  **public** **void** setTitle(String newTitle) { title = newTitle; } // sets title  **public** **void** setSalary( **int** newSalary) { salary = newSalary; } // sets salary  // accessors  **private** String getName( ) { **return** name; } // retrieves salary  **public** String getDepartment( ) { **return** department; }// retrieves department  **public** String getTitle( ) { **return** title; }// retrieves title  **public** **int** getSalary( ) { **return** salary; }// retrieves salary  // toString  **public** String toString( ) { **return** name + "\n" + department + "\n" + title + "\n" + salary;}  } |
| Ravi.setSalary(10); ~~Ravi.salary = 10;~~ |

Proving Encapsulation works

* Now that data member are private, I SHOULD NOT have direct access
* I have to call functions in order to change them
  + They can validate data, validate the user, etc…

|  |
| --- |
| Data members were *public* |
| **public** **class** Employee {  **public** String name;  **public** String department;  **public** String title;  **public** **int** salary;  } |
|  |

|  |
| --- |
| Data members are *private* |
| **public** **class** Employee {  **private** String name;  **private** String department;  **private** String title;  **private** **int** salary;  } |
|  |

Encapsulation – Instance (main) setup

* remember
  + an instance is created in the main/driver
  + instance values are set/returned in the driver

|  |
| --- |
| **In the main()** |
| …  Employee EM001 = **new** Employee();  EM001.setName("Mr. Lupoli");  EM001.setDepartment("Computer Science");  EM001.setTitle("Assistant Professor");  EM001.setSalary(-1);  System.out.println(EM001); // uses toString method in class  // what do you think the LAST line will print? |

1. Change all of your data members in Employee to PRIVATE.
2. Delete all instances created in the main().
3. Create 2 ***instances*** using the Employee class, of yourself, and a friends’ info (can be fake info) using the **new** structure: ***(DO NOT RECREATE THE CLASS!!)***

Creating non-setter/getter methods

* You CAN create your own!!
* Needs to be completed in these steps

1. Create the function WITHIN the class
2. Call the function WITHIN the driver

|  |
| --- |
| Custom Object Methods |
| Class |
| **class** Employee  {  **private** String name;  **private** String department;  **private** String title;  **private** **int** salary;  // mutators  **public** **void** setName(String newName)  {  name = newName;  } // set name  **public** **void** setDepartment(String newDP){department = newDP;}// sets department  **public** **void** setTitle(String newTitle) { title = newTitle; } // sets title  **public** **void** setSalary( **int** newSalary) { salary = newSalary; } // sets salary  **public** **void** setSalaryRaise(**double** percentage)  {  salary += salary \* percentage;  }    // accessors  **private** String getName( ) { **return** name; } // retrieves salary  **public** String getDepartment( ) { **return** department; }// retrieves department  **public** String getTitle( ) { **return** title; }// retrieves title  **public** **int** getSalary( ) { **return** salary; }// retrieves salary  // toString  **public** String toString( ) { **return** name + "\n" + department + "\n" + title + "\n" + salary;}    } |

|  |
| --- |
| Driver |
| **public** **class** Driver {  **public** **static** **void** main(String[] args) {  Employee EM001 = **new** Employee();  EM001.setName("Mr. Lupoli");  EM001.setDepartment("Computer Science");  EM001.setTitle("Assistant Professor");  EM001.setSalary(100);  System.***out***.println(EM001);  EM001.setSalaryRaise(0.02);  // let's see if the raise stuck!!  System.***out***.println(EM001);  }  } |

**What can you put into a method??**

* Anything you have already learned!!
  + Loops
  + Arrays
  + If-else
  + Variables
  + Etc…

**See how a method works (Mechanics)**

* code literally makes a jump in the program
* function then returns back to where it was called

|  |  |
| --- | --- |
| **Method Mechanics** | |
| Employee Class | Employee Driver |
| **class** Employee  {  **private** String name;  **private** String department;  **private** String title;  **private** **int** salary;  // mutators  **public** **void** setName(String newName)  {  name = newName; ➋  } // set name➌ | import java.util.Random;  import java.util.Scanner;  public class helloWorld  {  static Scanner sc = new Scanner(System.in);  public static void main(String[] args)  {  Employee EM001 = new Employee();  EM001.setName("Mr. L"); ➊    }  } |

**Parameters**

* ***what does a method need for it to work?***
  + Example, finding the hypotenuse of a right triangle needs:
    - side A length
    - side B length
* need to ***pass*** both values into the function if you wish to find the hypotenuse
* the parameter list can be reused over and over again with DIFFERENT values
  + called aliases

**Review of Hypotenuse and triangles**

c2 = a2 + b2 (c being the hypotenuse)

|  |  |  |  |
| --- | --- | --- | --- |
|  | length |  |  |
|  | 62 + 82 = c2  36 + 64 = c2  100 = c2  c = √100  c = 10 |  |  |

**Hypotenuse function using parameters**

|  |
| --- |
| **The function** |
| public double getHypotenuse(int a, int b)  {  return Math.sqrt(a\* a + b \* b);  }  What datatype will this function return?  What variables are parameters? |
| **The call to that function** |
| Triangle example = new Triangle();  double answer1 = example.getHypotenuse(8,6);  double answer2 = example.getHypotenuse(9, 12);  double answer3 = example.getHypotenuse(16,30);  // Please notice that “a” and “b” are reused and given new values EACH time!! |
| Receiving parameters from the user |
| // user input values  System.out.println( “give 2 values for a right triangle”);  int x = sc.nextInt();  int y = sc.nextInt();  double answer4 = example.getHypotenuse(x,y);  // Please notice that “a” and “b” are reused and given new values EVEN HERE!!  Why will x and y work?? |

**Methods with Parameters in Employee**

* fitting an example of a custom function with multiple parameters and an Employee
* remember that methods, can access values from the instance (member variables) not just from parameters

|  |
| --- |
| Member functions with parameters |
| **public** **void** printPayCheck(**double** hours, **double** taxPercentage)  {  System.*out*.println("----------- You should receive in your check --------");  **double** pay = hours \* wage \* (1.00-(taxPercentage/100));  System.*out*.println("" + fmt.format(pay));  }  // wage was a member from the class  // hours and taxPercentage were values passed in as parameters |
| Employee e = **new** Employee("John Adams", "Boston, MA", 2, 2400.00);  e.setWage(12.50);  e.printPayCheck(35, 33); |

**Alias**

* values given temporary name ***HAVE TO SHARE THE*** same data type
* items and functions asks for:
  + how many? (parameters)
  + what type are each?
    - THEY DO NOT ASK FOR A NAME!!!

|  |  |
| --- | --- |
| public double getHypotenuse(int a, int b)  {  return Math.sqrt(a\* a + b \* b);  } | // how many parameters does the function need to work?  // what TYPES does it need |
| public void function(int x, double y, char z)  {  …  } | // how many parameters does the function need to work?  // what TYPES does it need |
| public char function (int x, int y, double z)  {  …  } | // how many parameters does the function need to work?  // what TYPES does it need |
| public void function (double x, double y, char z)  {  …  } | // how many parameters does the function need to work?  // what TYPES does it need |
| public double function (int x, double y, char []z)  {  …  } | // how many parameters does the function need to work?  // what TYPES does it need |