**Notes to build Acme Inc system**

CJS Notes

Things to add

Constants private static final double EMPLOYEE\_DISCOUNT = .1;

Add workerType, set to enum fullTime, partTime, contractor, seasonal

Override equals method to see if 2 objects of the same class are equal

@Override

public boolean equals(Object obj)

{

Car c = (Car)obj;

return (this.color.equals(c.color) && (this.numberOfDoors == c.numberOfDoors);

}

Create workspace C:\TTS\Acme\_Inc

Create package com.acme

Create AcmeMain

**package** com.acme;

**public** **class** AcmeMain {

**public** **static** **void** main(String[] args) {

Controller controller = **new** Controller();

controller.runDemo();

}

}

Create Controller

**package** com.acme;

**public** **class** Controller {

private Customer customer = **new** Customer();

private Employee employee = **new** Employee();

**protected** **void** runDemo() {

loadData();

showStatus();

}

// private in order to do “Data hiding”

**private** **void** loadData() {

// Set Values

customer = **new** Customer("Joe","Namath","joe@gmail.com", "123");

employee = **new** Employee("Danny","Devitto", "DD@Yahoo.com","A123",123213.23);

}

**private** **void** showStatus() {

System.out.println(customer.toString());

System.out.println(employee.toString());

}

}

Show how you can right click Quick Type Hierarchy

Create package com.acme.bo // business objects

Create class Customer

**package** com.acme.bo;

**public** **class** Customer {

**private** String firstName;

**private** String lastName;

**private** String email;

**private** String customerId;

// Right click -> Source -> Generate Setters and Getters, these are access modifiers

@Override //annotation

**public** String toString() {

StringBuffer sb = **new** StringBuffer();

sb.append("Name: " + getFirstName() + " " + getLastName() + "\n");

sb.append("email: " + getEmail() + "\n");

sb.append("Customer Id: " + getCustomerId() + "\n";

**return** sb.toString();

}

}

Create instance of Customer in loadData(), write to console in showStatus()

// run in AcmeMain

(Break)

Add a constructor to Customer

**public** Customer (String firstName,String lastName, String email, String id) {

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.email = email;

**this**.setCustomerId(id);

}

// add an overloaded method as alternate constructor, also the default constructor

**public** Customer () {

**}**

// Rerun from AcmeMain

**Create Employee**

Copy Customer to Employee (or have students create employee as a lab)

**private** String employeeId;

**private** **double** salary;

Modify Constructor

**public** Employee (String firstName,String lastName, String email, String id, **double** salary) {

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.email = email;

**this**.setEmployeeId(id);

**this**.setSalary(salary);

}

Modify toString()

"Employee Id: " + getEmployeeId() + "\n" + "Salary: " + getSalary()+ "\n";

Add to loadData and ShowStatus

---------------------------

**Add some validation (Should be about 1 hr to here)**

//Validation – Carlson Companies

**public** **void** setEmail(String email) {

**if**(!email.contains("@") || !email.contains(".")) {

**throw** **new** RuntimeException("Email must have a @ and a . character [" +

email+ "]");

}

**this**.email = email;

}

/\*\*

\* Customer id must be numeric

\* **@param** customerId -

\*/

**public** **void** setCustomerId(String customerId) {

**try** {

Integer.*parseInt*(customerId);

} **catch** (NumberFormatException e) {

**throw** **new** RuntimeException("Customer ID must be numeric [" + customerId +"]");

}

**this**.customerId = customerId;

}

**Lab: Add String SSN to all your employees, Add validation. Include in toString(), include in constructor**

// add this to Employee object

**private** String ssn;

// generate getter and setter

**public** **void** setSsn(String ssn) { // add validation

**try** {

Integer.*parseInt*(sSN);

} **catch** (NumberFormatException e) {

**throw** **new** RuntimeException("SSN must be numeric [" + ssn +"]");

}

**if** (ssn.length() != 9) {

**throw** **new** RuntimeException("SSN must be 9 digits [" + ssn +"]");

}

This.ssn = ssn;

}

**public** Employee (String firstName,String lastName, String email, String id, **double** salary, String ssn) {

**super**(firstName, lastName, email);

**this**.setEmployeeId(id);

**this**.setSalary(salary);

**this**.setSsn(ssn);

}

Modify Controller to add ssn to constructor

**Create and show List of employees and customers (Maybe show customers and have them do employees)**

**public** **class** Controller {

List<Customer> customers = **new** ArrayList<Customer>();

List<Employee> employees = **new** ArrayList<Employee>();

**protected** **void** runDemo() {

loadData();

showStatus();

}

**private** **void** loadData() {

Customer cust1 = **new** Customer("Joe","Namath","joe@gmail.com", "123");

customers.add(cust1);

Customer cust2 = **new** Customer("Michael","Jordan","dunk@gmail.com", "124");

customers.add(cust2);

Customer cust3 = **new** Customer("Hank","Aaron","slugger@gmail.com", "125");

customers.add(cust3);

Employee emp1 = **new** Employee("Danny","Devitto", "DD@Yahoo.com","A123",123213.23,"142231234");

employees.add(emp1);

Employee emp2 = **new** Employee("Arnold","Schwarzeneger", "Arnie@Yahoo.com","B234",223213.23,"321323214");

employees.add(emp2);

}

**private** **void** showStatus() {

**for**(Customer customer:customers) {

System.***out***.println("----------------------------");

System.***out***.println(customer.toString());

}

System.***out***.println("==================================");

**for**(Employee employee:employees) {

System.***out***.println(employee.toString());

System.***out***.println("----------------------------");

}

}

}

// Run main and see output

**Create Person (explain why we need to refactor)**

package com.acme.bo;

**public** **abstract** **class** Person{ // this fails if abstract >> Person person = **new** Person();

private String firstName;

private String lastName;

private String email;

// generate getters and setters source -> generate getters and setters

public Person (String firstName,String lastName, String email) {

this.setFirstName ( firstName);

this.setLastName ( lastName );

this.setEmail ( email);

}

@Override

public String toString() {

StringBuffer sb = new StringBuffer();

sb.append("Name: " + firstName + " " + lastName + "\n");

sb.append("email: " + email + "\n");

return sb.toString();

}

// generate getters and setters

}

Modify Employee **extends** Person

// constructor

**public** Employee (String firstName,String lastName, String email, String id, **double** salary) {

**super**(firstName, lastName, email);

**this**.setEmployeeId(id);

**this**.setSalary(salary);

}

// toString

@Override

**public** String toString() {

String outPutString = **super**.toString()+ "Employee Id: " + getEmployeeId() + "\n" + "Salary: " + getSalary()+ "\n";

**return** outPutString ;

}

Modify Customer **extends** Person

//constructor

**public** Customer (String firstName,String lastName, String email, String id) {

**super**(firstName, lastName, email);

**this**.setCustomerId(id);

}

//toString

@Override

**public** String toString() {

String outPutString = **super**.toString();

**return** outPutString + "Customer Id: " + getCustomerId() + "\n";

}

//Show how you can right click Quick Type Hierarchy to see customer extends person

//Show what methods are visible on Customer, will include methods from Person

**Create and display persons list in Controller**

List<Person> persons = **new** ArrayList<Person>();

**private** **void** loadData() {

persons.addAll(customers);

persons.addAll(employees);

**private** **void** showStatus() {

**for**(Person person:persons) {

System.***out***.println("----------------------------");

System.***out***.println(person.toString()); // shows polymorphism

}

**CJS: Start up here next week**

**Add Orders**

// Order Processor Interface

**package** com.acme;

**public** **interface** OrderProcessor {

/\*\*

\* Add an order

\*/

**public** **void** addOrder(**double** orderTotal);

/\*\*

\* Returns a string of total purchase amount for an order processor

\*/

**public** String getTotalPurchaseAmount();

/\*\*

\* Returns the discount for the order processor

\*/

**public** **double** getDiscount();

}

// Update Person Object

Person **implements** OrderProcessor

List<Double> orders = **new** ArrayList<Double>();

**public** **void** addOrder(**double** orderTotal) {

orders.add(orderTotal);

};

**public** String getTotalPurchaseAmount() {

**double** totOrders = 0.0;

**if** (**null** != orders) {

**for**(**double** price:orders) {

totOrders += price;

}

}

// deduct discount

totOrders = totOrders \* (1 – getDiscount());

**return** **this**.getFirstName() + “ “ + **this**.getLastName() + “ Order total = “ + totOrders;

};

// Update Employee

**public** **double** getDiscount() {

**return** .1;

}

// Update Customer

**public** **double** getDiscount() {

**return** .05;

}

// Update Controller

List<OrderProcessor> allOrders = **new** ArrayList<OrderProcessor>();

// Shortcut to add orders

In loadData()

allOrders.addAll(persons);

// add some orders

**for**(OrderProcessor op:allOrders) {

op.addOrder(100);

op.addOrder(100);

op.addOrder(100);

op.addOrder(100);

op.addOrder(100);

}

**private** **void** showStatus() {

**for**(OrderProcessor orderProcessor:allOrders) {

System.***out***.println(orderProcessor.getTotalPurchaseAmount());

}

// Output should be as follows, polymorphism.

Joe Namath Order total = 475.0

Michael Jordan Order total = 475.0

Hank Aaron Order total = 475.0

Danny Devitto Order total = 450.0

Arnold Schwarzeneger Order total = 450.0

**Enums**

// Create Enum

**package** com.acme.enums;

**public** **enum** WorkerTypeEnum {

***FULL\_TIME***,

***PART\_TIME***,

***CONTRACTOR***,

***SEASONAL***

}

// Add to Employee.java

**private** WorkerTypeEnum workerTypeEnum;

**public** WorkerTypeEnum getWorkerTypeEnum() {

**return** workerTypeEnum;

}

**public** **void** setWorkerTypeEnum(WorkerTypeEnum workerTypeEnum) {

**this**.workerTypeEnum = workerTypeEnum;

}

// Add to Person object

@Override

**public** String toString() {

StringBuffer sb = **new** StringBuffer();

sb.append(**this**.getTotalPurchaseAmount()+ "\n");

sb.append("Name: " + firstName + " " + lastName + "\n");

sb.append("email: " + email + "\n");

sb.append("Worker Type: "+ **this**.getWorkerTypeEnum()+ "\n");

**return** sb.toString();

}

Create StringUtils.hasSpecialChars()

Create package com.acme.util

**package** com.acme.util;

**public** **class** StringUtils {

**public** **static** **boolean** hasSpecialChars(String inString) {

**char**[] inArray = inString.toCharArray();

String specChars = "~!@#$%^&\*()\_-+=:;/?<.>,";

**for** (**char** c:inArray) {

**if** (specChars.indexOf(c) > 0) **return** **true**;

}

**return** **false**;

}

}

**public** **void** setFirstName(String firstName) {

**if** (StringUtils.*hasSpecialChars*(firstName)) {

**throw** **new** RuntimeException("Name must not have special characters [" + firstName+ "]");

}

**this**.firstName = firstName;

}

// Possible lab, create ValidationUtils.validateEmail()