CS102 – Algorithms and Programming II
Lab Programming Assignment 4
Fall 2017

ATTENTION:

- Feel free to ask questions on Moodle on the Lab Assignment Forum.
- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention:

  CS102_SecX_Asgn3_YourSurname_YourName.zip
- Replace the variables "YourSurname" and "YourName" with your actual surname and name and X with your Section id (1 or 3).
- Upload the above zip file to Moodle by the deadline before the lab (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TA during the lab. You will resubmit your code once you demo your work to the TA.

GRADING WARNING:

- Please read the grading criteria provided on Moodle.

Q1 [100 p] In this assignment, you are going to complete your Cargo program by introducing new delivery mechanics through additional classes.

- **The following classes will remain the same from the last assignment:** *Customer, Locatable, Item, Delivery, Mail, Package.*
- **Person is not abstract in this assignment.**
- **Vehicle:** Drivers use Vehicles to deliver the packages to their relative receiving **Customers**. Every vehicle has a constant speed, capacity and weight limit depending on its type. Vehicles are instantly returned to **Company** after each single delivery, so the delivery of each package, mail always starts from **Company**. Add **Vehicle**(abstract) and its two children **Motorcycle** and **Truck. Vehicle** also *implements* **Locatable.**
  - **Vehicle:**
    - **Properties:**
      - **Int vehicleNo:** The id of the vehicle
      - **Delivery[] deliveries:** Array of deliveries
      - **Int NumOfDeliveries:** Current number of deliveries
      - **Double currentWeight:** Current total weight of deliveries
      - **Double speed:** Speed of the vehicle (constant)
    - **Methods:**
      - **setSpeed(value),getSpeed()**
      - **int getNumOfDeliveries()**

- **double getCurrentWeight()**
- **int getVehicleNo();**
- **void loadDelivery(Delivery delivery):**
  - Loads the given delivery into its relative array.
- **static double calculateDistance(Locatable source,Locatable target):**
  - Calculates the Euclidian distance between two Locatable objects. Mind static keyword.
- **String abstract toString():**
- **abstract double getWeightLimit();**
- **abstract int getDeliveryLimit();**
- **abstract String drive();**

- **Truck:** The slow type of a vehicle, but it can carry a lot.
  - **Constants:**
    - **int TRUCK_CAPACITY:**
    - **double TRUCK_SPEED:**
    - **double TRUCK_WEIGHT_LIMIT:**
  - **Methods:**
    - **Truck (int vehicleNo):**
    - **String drive():** Returns a string containing the delivery information of all packages, mails. Displays type, no, sender and receiver info (name and loc), and time it took to deliver it from Company's location, for each delivery.
      - *Time took*: Distance between (receiver, company) / speed
      - After delivery, Motorcycle resets deliveries.
    - **int getDeliveryLimit():**
    - **double getWeightLimit():**
    - **String toString():** Type of this vehicle, its no and number of deliveries.
- **Motorcycle:** Faster than **Truck,** yet it cannot carry as much. Different from **Truck,** it doesn't return the names of the customers upon delivery.
  - **Constants:**
    - **int MOTORCYCLE_CAPACITY:**
    - **double MOTORCYCLE _SPEED:**
    - **double MOTORCYCLE _WEIGHT_LIMIT:**
  - **Methods:**
    - **Motorcycle (int vehicleNo):**
    - **String drive():**Returns a string containing the delivery information of all packages, mails. Displays type, no, sender

and receiver info (loc only), and time it took to deliver it from Company's location, for each delivery.
    - *Time took*: Distance between (receiver, company) / speed
    - After delivery, Motorcycle resets deliveries.
- **int getDeliveryLimit():**
- **double getWeightLimit():**
- **String toString():** Type of this vehicle, its no and number of deliveries.

- **Employee:** In this assignment, the Employee class is **abstract** as they will specialize on their job at the Company. Add two new classes that extends **Employee** class; **CustomerService** and **Driver.** (They will be concrete classes, **not abstract**) Employee class still extends **Person** class.
    - **CustomerService:** Handles the packaging of the items given by the customers to the **Company.** But, it has a limited amount of jobs it can handle at a single time and only finishes when the packages are delivered to the Customers. When **CustomerService** has reached its job limit, it won't be available.
        - **Properties:**
            - ***MAX_JOBS (int):*** maximum number of jobs the customer service employee can handle at a time (must be constant)
            - ***currentJobs (int):*** current number of jobs
        - **Methods:**
            - ***CustomerService(int employeeId, string name):*** Constructor
            - ***boolean getAvailability():*** Returns true if the customer service employee is available
            - ***Delivery addJob(Item item, Customer sender, Customer receiver, int packageNo):*** By considering the weight of the item, creates a **concrete** Delivery object (Mail or Package)
            - **void finishJobs():** Prints out the number of jobs completed and resets **currentJobs**.
            - **String toString():** Information about the class
    - **Driver:** Uses **Vehicle** to deliver the packages given by the Company. Can only be associated to a single Vehicle. (*Assume the user never tries to assign multiple drivers to a single vehicle*). Driver is also responsible for loading the deliveries onto associated vehicle, while respecting its limits.
        - **Properties:**
            - ***Vehicle vehicle:*** The assigned vehicle to this driver object

        - ***Methods:***
            - ***Driver (int employeeNo, String name):*** Constructor
            - ***void assignVehicle(Vehicle vehicle):*** Assigns the given vehicle to this driver
            - ***boolean giveCargo(Delivery delivery):*** Load the given delivery onto the vehicle, if it has enough capacity and doesn't exceed its weight limit. Returns true if the loading was successful
            - ***void deliver():*** Drive the vehicle and print how many deliveries has been completed. If no vehicle has been assigned, print relative info.
            - **String toString():** Information about the class
    - **Employee (abstract):** This class has been deducted to following methods as packaging and delivery methods are moved to child classes.
        - **Properties:**
            - ***Double salary:*** The salary of the employee
            - ***Int employeeNo:*** *The no of the employee*
        - **Methods:**
            - ***Employee(int employeeNo,String name):*** Constructor
            - ***void adjustSalary(double value):*** Adds the given value to the salary
            - ***String abstract toString()***

- **Company:** Company class will be updated, to be compatible with new classes and their functions. Now, **Company** holds the deliveries which are packed by **CustomerService**, and loaded onto **Vehicle** by the **Driver** it is assigned to. Certain methods return Boolean values, indicating if the related operation was successful or not. Company still implements **Locatable.**
    - **Properties**
        - **final int** EMPLOYEE_CAPACITY and VEHICLE_CAPACITY**;**
        - **Array** to hold Employee objects and another one for Vehicle objects
        - An **ArrayList** to hold Delivery objects and another one for Customer objects
        - **Int** numOfEmployees, numOfVehicles
        - **Int** packageNo, employeeNo, vehicleNo (each starting from 0, incrementing each time a new object has been added to Company. They are used as ID's)
        - Properties from Locatable class
    - **Methods**

- **Company(int posX,int posY):** Constructor
- **void addCustomer (Person customer)**
- **boolean hireEmployee(Person candidate, String type):** Hires the given **Person** as the type of Employee. The correct types are “CustomerService” or “Driver”, rest should cause the operation to be unsuccessful.
- **boolean createDeliverable (Item sendItem,Customer sender, Customer receiver):** Creates a delivery object from the item with sending info. Uses an available customer service employee for packaging.
- **boolean addVehicle(String type):** Adds the given type of vehicle to the Company. The correct types are “Motorcycle” or “Truck”, rest should cause the operation to be unsuccessful.
- **boolean assignVehicle(int employeeIndex, int vehicleIndex):** Given the driver and vehicle index (their indices in their respective arrays), assign the vehicle to the driver. As the drivers are in the employee’s array, the given index should be checked if it is a driver.
- **boolean loadDelivery(int deliveryNo, int employeeIndex):** Loads the delivery to the vehicle of the given driver.
    - The deliveryNo is not an index (as they are held in an ArrayList), so you need to search for the delivery with the corresponding index. Don’t forget to remove the delivery from the list if it has been successfully loaded.
    - Returns false if the operation fails.
    - (Hint: Use **giveCargo** method of Driver to load the package onto Vehicle)
- **void deliverPackages():** Drivers deliver their packages and Customer Service Employees finish their jobs.
- **String toString():** List and print all the information related to the Company. Includes deliveries, customers, vehicles, drivers and customer services.

- Update your **CompanyTester** into a simple menu that enables you to test the program. An example result (non-menu) is shown below:

```
            //Create company
            Company comp = new Company(0, 0);

            //Create people
            Person pers1 = new Customer("Antonio");
            comp.addCustomer(pers1);
            pers1.setPos(4, 3);
            Person pers2 = new Customer("Jack");
            comp.addCustomer(pers2);

            //Hire them
            Person pers3 = new Person("Baby Driver");
            comp.hireEmployee(pers3, "Driver");

            Person pers5 = new Person("Han Solo");
            comp.hireEmployee(pers3, "Driver");

            Person pers4 = new Person("Brad Pitt");
            comp.hireEmployee(pers4, "CustomerService");

            //Create items
            Item item1 = new Item(5,"A plate");
            Item item2 = new Item(0.01,"TOP SECRET");

            //Send between customers
            ((Customer)comp.customers.get(1)).sendItem(comp, item1,
(Customer)comp.customers.get(0));
            ((Customer)comp.customers.get(0)).sendItem(comp, item2,
(Customer)comp.customers.get(1));

            //Add vehicles
            comp.addVehicle("Truck");
            comp.addVehicle("Motorcycle");

            //Assign them to drivers
            comp.assignVehicle(0, 1);
            comp.assignVehicle(1, 0);

            //Load deliveries
            comp.loadDelivery(1, 0);
            comp.loadDelivery(0, 1);

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

            //Deliver
            comp.deliverPackages();

            System.out.println(comp.toString());
```

```
Company at 0 - 0
Employees: 0 Driver [vehicle=Motorcycle No:1 with 1 deliveries]
1 Driver [vehicle=Truck No:0 with 1 deliveries]
2 CustomerService [currentJobs=2, MAX_JOBS=5]
Vehicles
0 Truck No:0 with 1 deliveries
1 Motorcycle No:1 with 1 deliveries
Customers
Customer [currentItem=null, Name=Antonio, X=4, Y=3]
Customer [currentItem=null, Name=Jack, X=0, Y=0]
Deliveries

Driver 0 had delivered 1 package(s) with motorcycle (1)
Mail - 1 has been delivered from 4-3 to 0-0 in 0.0secs

Driver 1 had delivered 1 package(s) With truck (0)
Package - 0 has been delivered from Jack at 0-0 to Antonio at 4-3 in 1.0secs

Customer service employee 2 had completed 2 task(s)
Company at 0 - 0
Employees: 0 Driver [vehicle=Motorcycle No:1 with 0 deliveries]
1 Driver [vehicle=Truck No:0 with 0 deliveries]
2 CustomerService [currentJobs=0, MAX_JOBS=5]
Vehicles
0 Truck No:0 with 0 deliveries
1 Motorcycle No:1 with 0 deliveries
Customers
Customer [currentItem=null, Name=Antonio, X=4, Y=3]
Customer [currentItem=null, Name=Jack, X=0, Y=0]
Deliveries
```

The UML diagram of the program