CS102- Algorithms and Programming II

Lab 04

Lab Objectives: Inheritance, Abstract Classes, Interfaces, Polymorphism

---

Notes:

- For all labs in CS 102, your solutions must conform to these CS101/102 style guidelines.
- Create a Java Project named Lab04. Put all of your classes in this project.
- Remember to include **javadoc comments** for each class and method.
- Upload your solution **as a single .zip file** to the Lab04 assignment by the end of your section's lab session in the week of March 8. You must use the following naming convention: Lab04_Surname_FirstName.zip where Surname is your family name and FirstName is your first name. You may upload multiple times; the last upload will be considered.

Please **reuse** the available methods as much as you can instead of repeating the same code in different methods.

In this assignment, you will implement a program that sends items between customers. A customer orders items and the company creates a package with those items and delivers it. The company has employees who are responsible for packing the items. The item is delivered as a mail or package depending on the item weight. When the package is delivered, its delivery information is printed on the screen. To implement this program, you will need the following interface and class hierarchy:

`Locatable` - An interface that has methods with the following signatures:

- `int getX()`
- `int getY()`
- `void setPos(int x, int y)`

`Item` - A class that creates an item that is converted to a **Delivery** first to be sent. This class should have the following instance variables:

- `double weight`
- `String content`

This class should also have the following methods:

- `Item(double weight, String content)`: this constructor initializes the weight and the content of an item
- Get methods for weight and content
- `String toString()`: returns the properties of an item

`Delivery` – An abstract class that creates deliveries. This class should have the following instance variables:

- `int packageNo`

- **Customer** sender
- **Customer** receiver

This class should also have the following methods:

- Delivery(**Customer** sender, **Customer** receiver, **int** packageNo): this constructor initializes the sender, receiver and package number of a delivery
- Get method for sender, receiver and packageNo
- **abstract double** getWeight()

Mail - A class that extends **Delivery**. Returns a constant weight and only contains the content of the respective item. This class should have the following instance variable:

- **String** content

This class should also have the following methods:

- Mail(**String** content, **Customer** sender, **Customer** receiver, **int** packageNo)
- **double** getWeight(): (Returns 0.1, independent from content)
- **String** toString(): returns the properties of the mail

Package - A class that extends **Delivery**. This class should have the following instance variable:

- **Item** packedItem

This class should also have the following methods:

- Package(**Item** content, **Customer** sender, **Customer** receiver, **int** packageNo)
- **double** getWeight():
- **String** toString(): returns the properties of the package

Person – An **abstract** class that implements **Locatable**. This class should have the following instance variable:

- **String** name
- **int** posX
- **int** posY

This class should also have the following methods:

- Person(**String** name, **int** x, **int** y): this constructor should initialize the name of the person and his/her position
- Person(**String** name): this constructor should initialize the name of the person and his/her position to 0
- Get and set method for name
- All methods from the **Locatable** interface.

Customer - A class that extends **Person**. This class should have the following instance variables:

- **Item** currentItem

This class should also have the following methods:

- Customer(**String** name)
- Get and set method for currentItem

- **boolean** `sendItem(`**Company** `company,` **Item** `item,` **Customer** `receiver)` **:** this method sends an item via given company to the receiver if an employee is available(returns false otherwise). `item` object is ignored if the customer already has an item to be sent.
- **String** `toString()`**:** returns the properties of the Customer

Employee - A class that extends **Person**. Responsible for packaging and delivery. This class should have the following instance variables:

- **final int** `MAX_JOBS = 5;`
- **int** `currentJobs; //Initialized as 0`
- **Delivery[]** `deliveries; //The undelivered packages, mails are held here.`
- **double** `salary`
- **int** `employeeNo`
- **boolean** `available`

This class should also have the following methods:

- `Employee(`**int** `employeeNo,` **String** `name)`
- **void** `adjustSalary(`**double** `value)` **:** this method adjusts the employee's salary by a given value.
- **boolean** `getAvailability()`**:** this method returns if the employee is available or not
- **void** `addJob(`**Item** `sendItem,` **Customer** `sender,` **Customer** `receiver,` **int** `packageNo)`**:** this method determines the type of item and converts it to either mail (weight <= 0.1) or package (weight > 0.1) and adds it to the array of deliveries.
- **void** `deliverPackages()`**:** this method prints the information of the delivered items and of the sender and receiver customer. This method also clears the array of deliveries.
- **String** `toString()`**:** returns the properties of the Employee

Company - A class that implements **Locatable**. This class should have the following instance variables:

- **final int** `EMPLOYEE_CAPACITY = 15`
- **Employee[]** `employees`
- **ArrayList<Customer>** `customers`
- **int** `numOfEmployees`
- **int** `employeeNo`
- **int** `packageNo`
- **int** `posX`
- **int** `posY`

This class should have the following methods:

- `Company(`**int** `x,` **int** `y)` : this constructor should set the position and initialize any instance variables you may have.
- All methods from the **Locatable** interface.

- **boolean** addEmployee(**Employee** candidate): returns true if the employee is successfully added
- **void** addCustomer (**Customer** customer): adds the given customer
- **boolean** terminateContract(**int** employeeIndex): returns true if the employee at the given index is deleted
- **boolean** createDeliverable(**Item** sendItem, **Customer** sender, **Customer** receiver): this method creates a deliverable object from the item if an employee is available and returns true, otherwise it returns false
- **void** deliverPackages(): Deliver all the packages via Employees and print the delivery information. Displays type, no, sender and receiver info (name and loc) for each delivery.
- **String** toString(): List and print all the information related to the Company. Includes deliveries, employees, and customers.

Finally, you need to implement a **CompanyTester** class to test the classes and the methods you have implemented, using a simple menu system. The test cases should contain the following:

- Create several items with different weights.
- Create at least 2 Customers and 2 Employees.
- Customers send items to each other. Have a case where the Customer couldn't send the item, because there was no available Employee.
- Get an employee busy by giving it the maximum amount of jobs.
- Deliver items between customers.