

## 1 Objective

- Use Violet to create UML diagrams.

## 2 Introduction

Tools are critical in doing any successful work. Software design and development are no exception. Without good tools the difficulty level of programming would be increased dramatically. Imagine how difficult it would be if you were still using teletype terminals, or punched cards to enter programs into a computer.

In today's lab, you will learn to use Violet which is a drawing tool to create UML diagrams that represent Java classes.

## 3 Create a Project for Today's Lab

Create an Eclipse project for today's lab, calling it lab04. Import the files from

`~csci204/2009-fall/student/labs/lab04`

Remember to import the files into the project's `src` directory. When you are finished importing, commit the lab to your subversion lab repository by right clicking on lab04 and selecting **Team** → **Share Project...**

You should have the following files.

- a fully implemented `BankAccount` class
- `BankAccount.class.violet` which is a UML diagram for the `BankAccount` class
- a `Person` class

Each of the classes contains a `main()` method that you will use to test it.

## 4 Exercise

In the rest of this lab, you will be testing the `BankAccount` class. You will also be creating and testing a new class `Address`. At each step you are asked to test the classes. You should use the `main()` method in each class as your test method. You are also asked to create a UML diagram for all the classes.

### 4.1 The BankAccount Class

Run the `main()` method in the `BankAccount` class. Observe the results. Examine the implementation for the `BankAccount` class. Pay special attention to the testing components in the `main()` method.

Examine the UML diagram for the `BankAccount` class by double clicking on `BankAccount.class.violet`. Examine the Java code and find the corresponding parts in the UML diagram. Close the UML diagram when you are finished.

### 4.2 The Person Class

Next, examine the class `Person`. This class doesn't compile because the `Address` class doesn't exist. When you have finished writing the `Address` class, you will test it by running its `main()` method.

### 4.3 The Address Class

Create a new class called `Address`. Here are some guidelines.

1. When you ask Eclipse to create the class for you, check the box that says **public static void main(String[] args)** so that it will generate a `main()` method for you. You will use this method for testing. You should also check the **Generate comments** box.
2. Add instance variables for street, city, state, and the zip code. All should be of type `String` except that the zip code should be an integer. Make the instance variables **private**. *Hint: You might want to look at the `Person` class to determine your field names!*
3. Use **Source** → **Generate Constructor using Fields...** to produce a constructor that will accept initial values for each of the instance variables. Check the box next to each instance variable in dialog that appears. Check the **Generate constructor comments** and **Omit call to default constructor super()** boxes. Fill in the comments.
4. Use **Source** → **Generate Setters and Getters...** to generate setters and getters for your class. Check all of the instance variable and check the **Generate method comments** box. Complete the comments for each method.
5. Write a `toString()` method for the class. It should produce a string containing a nicely formatted address.

6. Add a `requestStatementConfirmation(Address addy)` method to the `BankAccount` class that prints a confirmation message to the console indicating that a statement for the account with its account number has been sent to the provided address.
7. Double click on `BankAccount.class.violet` to open the UML diagram for the `BankAccount` class. In the project window, drag `Address.java` into the UML diagram. A diagram for `Address` will appear. Save the file and close it.

Add code to the `main()` method in the `Address` class to test your methods. Run your tests. If you are unsure about how to do this, look at the `main()` method in `Person`.

#### 4.4 Remove Errors

Once you have completed the `Address` class, everything should compile. If there are any remaining warnings or errors, correct them. Run the `main()` method in `Person` and make sure it is working properly.

#### 4.5 Complete the UML Diagram

Double click on the UML diagram again and drag `Person.java` into the window. A diagram for `Person` should appear.

Complete the diagram by indicating the relationships between the classes. You may need to move the boxes around to get the diagram to look nice.

You can find more information about Violet here (<http://violet.sourceforge.net/>). Click on the *User guide* link.

### 5 Upon Completion

When you are finished, commit a final version of the lab to your subversion repository.

### 6 For Later Usage

In this lab, you edited an already existing UML diagram with Violet. For most projects, you will start the UML before you begin coding. In order to begin a blank UML diagram, click on your project name and select **File** → **New** → **Other...** →

**Violet UML Editor** → **Class Diagram**. Pick a location inside your project (anywhere will do) and a name for your UML diagram. Eclipse will then bring up a blank UML diagram. You can use the diagram tool, Class, to create each new class. Right clicking on a UML class allows you to set its name, member data and methods. I strongly encourage you to use Violet, or some other diagram/workflow tool, to create UML diagrams for your class projects. It will take less time than building them from scratch and the output will be well formatted and easy to look at.