marking-sheet.txt

18

```
==============================================================
marking cits1001 project 1 for ngj04 at 2017-04-11 16:55
==============================================================

=== Submission and Compilation /2 ============================= 2

Person.java file submitted
AddressBook.java file submitted
Submitted Person has 00103 lines of code and 00064 non-comment lines
Submitted AddressBook has 00058 lines of code and 00040 non-comment lines
Person compiled successfully
AddressBook compiled successfully                                  2
TestPerson compiled successfully, signatures OK
TestAddressBook compiled successfully, signatures OK

=== JUnit tests (Correctness) /8 ============================== 7

TestPerson tests
TestAddressBook tests                                              7

Tests run: 27,  Failures: 1
OK (9 tests)
OK (3 tests)

Minor errors (each 1 mark off)

Major errors (each 2 mark off)


=== Clarity and Design / 10 ================================== 9


Student details included in Javadoc headers: Person and AddressBook   2

Code is neatly laid out and indented, consistent bracketing, lines no longer than 80
    characters                                                     2

Variables given appropriate names.                                 2

Helper methods and appropriate method reuse applied (e.g. addPerson uses findPerson,
    getActivityScore in Person)                                    1

Appropriate structures chosen (for-each loops, if without empty branches etc)   2


=== Extension /2 ============================================== 0

Extension.pdf not submitted.                                       0
```

feedback-tests.txt

```

JUnit version 4.12
.........
Time: 0.012

OK (9 tests)

JUnit version 4.12
...
Time: 0.008

OK (3 tests)

JUnit version 4.12
.................E...could not add person
.......
Time: 0.024
There was 1 failure:
1) b_testForInternedStrings(TestPersonMinorError)
org.junit.ComparisonFailure: first name should be unchanged expected:<[bob]> but was:<[]>

FAILURES!!!
Tests run: 27,  Failures: 1
```

```java
    /**
     * Set the person's surname unless the parameter is an empty string.
     *
     * @param surname A string of the person's new surname.
     */
    public void setSurname(String surname) {
        if(surname != ""){
            this.surname = surname;
        }
    }

    /**
     * Return the person's mobile phone number
     *
     * @return A string of the person's mobile number.
     */
    public String getMobile() {
        return mobile;
    }

    /**
     * Set the person's mobile phone number
     * unless the parameter is an invalid string.
     * A string is a valid mobile phone number if every character in it is a digit from 0 to 9.
     *
     * @param mobile A string of the person's new mobile number.
     */
    public void setMobile(String mobile) {
        if((mobile != "")&&(mobile != null)){
            for(char c: mobile.toCharArray()){ //search through each letter in mobile for non-digits
                if(!Character.isDigit(c)){
                    return;
                }
            }
            this.mobile = mobile;
        }
    }

    /**
     * Return the person's email address
     *
     * @return A string of the person's email address.
     */
    public String getEmail() {
        return email;
    }

    /**
     * Set the person's email address
     * unless the parameter is an empty sting
     *
     * @param email A string of the person's new email address.
     */
    public void setEmail(String email) {
        if(email != ""){
            this.email = email;
        }

    }
```

```
                }
            }
        }
        return null;
    }

    /**
     * Find the most social person in the address book.
     *
     * @return An object of class Person with the highest Social Activity Level.
     *          If two or more people have the same highest social media activity level,
     *          findMostSocial will return the first it finds. findMostSocial searches
     *          contacts sequentially starting from the first added contact.
     */
    public Person findMostSocial() {
        if(contacts.size() != 0){
            Iterator<Person> it = contacts.iterator();
            Person i = it.next(); //skip first person.
            Person mostSocial = i;
            int highLevel = i.getTotalActivityLevel(); //getTotalActivityLevel is a new method in Person.

            while(it.hasNext()){ //check through contacts for a higher social person.
                i = it.next();
                int iLevel = i.getTotalActivityLevel();
                if(iLevel > highLevel){
                    highLevel = iLevel;
                    mostSocial = i;
                }
            }
            return mostSocial;
        }
        return null;
    }
}
```