**Lab**​​**3**​​**-**​​**Build**​​**the**​​**First**​​**Java**​​**Application**

This ​ lab​ ​ contains​ ​ in-class​ ​ exercises​ ​ related​ ​ ​to ​ ​class ​ design,​ ​ ​abstract ​ class,​ ​ interface,​ inheritance, ​ and​ ​ array​ ​ operation.​

**Task** ​ **1:**​​Develop ​ an​ ​ abstract​ ​ class​​Person

Create ​ an​ ​ abstract​ ​ class​ ​ named​ ​Person ​​which ​ has​ ​ three​​ fields​​ to​ ​ hold​ ​ the​ ​ info​ ​ about​ ​ a​ person, ​ namely:​ ​ *name,*​​*last*​​*name*​​*and*​​*email*​​*address*​.

The ​ class​ ​ also​ ​ has​ ​ an​ ​ abstract​ ​​method: ​public ​​abstract​ ​void ​​doing();

**Task** ​ **2:**​​Develop ​ an​ ​ Interface​ ​Talkable

Create ​ an​ ​ Interface​ ​Talkable ​ with​ ​ a​ ​ method​ ​void​ ​say()

**Task**​​**3:**​Develop​classes: ​Student​,​ Professor ​ and​ ​Staff

Create ​ classes​ ​Student,​ ​Professor ​ and​ ​Staff ​ which​ ​ are​ ​ subclasses​ ​ ​of ​ the​ ​Person ​ class​ and ​ implement​ ​ the​ ​ interface​ ​Talkable.​

The ​ three​ ​ classes​ ​ implement​ ​ the​​ doing​ ​()​ method​ ​ by​ ​ printing​ ​ out​ ​ message​ ​ ​on

correspondingly ​ “​ *Studying*​ ​​*as*​​*student!”,*​​*“Teaching*​​*as*​​*professor!*​” ​ ​and ​ “​ *Working*​ ​​*as*​​*staff!*​”.

Add ​ at​ ​ least​ ​ one​ ​ field​ ​ to​ ​ each​ ​ of​ ​ the​ ​ three​ ​ classes,​ ​ e.g.​ ​ “program”​ ​ for​ ​ Student​ ​ class​​, “office” ​ for​ ​ Professor​ ​ class,​ ​ and​ ​ ​“title” ​ for​​ Staff​​ ​class.​ Each​ ​class​ ​ should​ ​ ​implement ​ the​ say​() ​ method​ ​ by​ ​ printing​ ​ messages​​ including​ ​ ​greeting, ​ full​ ​ name,​ ​ and​ ​ the​ ​ info​ ​ of​ ​ added​ field ​ above.​ ​ For​ ​ example,​ ​ ​ “ ​*Hello!*​​*My*​​*name*​​*is*​​*John*​Smith*,*​​​*and*​​*I'm*​​*in*​​*CPD*​​*program*​.”.

**Task**​​**4:**​Develop​class:​Meeting

Create ​ class​ ​Meeting ​ which​ ​ has​ ​ only​ ​ the​ ​main ​ method​ ​which ​ is​ ​ the​​ entry​ ​ to​ ​ run​​ the​ program. ​ Implement​ ​ the​ ​ following​ ​ in​ ​ the​ ​ ​main ​ method:​

Create ​ two​ ​ objects​ ​ for​ ​ each​ ​ class​​ of​ ​ Student,​ ​Professor,​ ​ and​ ​Staff ​ -​​ 6​ ​ objects​ ​ in​ ​ total.​ Create ​ an​ ​ array​ ​ of​ ​Person ​ with​ ​ 6​ ​ ​array ​ elements:​

Person[]​ ​working​ ​=​ ​new​ ​Person[6];

Initialize ​ the​ ​ array​ ​ with​ ​ the​ ​ 6​ ​ ​objects. ​ ​Use ​ for-loop​ ​ ​to ​ print​​ ​out ​ the​ ​ info​ ​ ​of ​ each​ ​ object​ ​ in​ the ​ array​ ​ by​ ​ calling​ ​toString()​ and​ ​doing() ​ methods. ​ Here​ ​ is​ ​ the​ ​ sample​​ output:​

@​ ​Wei​ ​Song,​ ​Jordan​ ​Anastasiade

Student​ ​[program=CPD,​ ​toString()=Person​ ​[firstName=John, lastName=Smith,​ ​email=jsmith@myseneca.ca]]

*Staff*​​*[title=Technical*​​*Support,*​​*firstName=Paul,*​​*lastName=Miller,*

*email=paul.miller@senecacollege.ca]*

*Working*​​*as*​​*staff!*​Studying​ ​as​ ​student!

Student​ ​[program=CPD,​ ​toString()=Person​ ​[firstName=Colin, lastName=Thomas,​ ​email=jsmith@myseneca.ca]] Studying​ ​as​ ​student!

Faculty​ ​[office=T1034,​ ​firstName=Jordan,​ ​lastName=Anastasiade, email=jordan.anastasiade@senecacollege.ca] Teaching​ ​as​ ​professor!

Faculty​ ​[office=T2099,​ ​firstName=Wei,​ ​lastName=Song, email=wei.song@senecacollege.ca] Teaching​ ​as​ ​professor!

Staff​ ​[title=Admin,​ ​firstName=Jack,​ ​lastName=Brown, email=jack.brown@senecacollege.ca] Working​ ​as​ ​staff!

Create ​ an​ ​ array​ ​ of​ ​Talkable ​ with ​ 6​ ​ ​array ​ elements:​

Talkable[]​ ​meeting​ ​=​ ​new​ ​Talkable[6];

Initialize ​ it​ ​ with​ ​ 6​ ​ objects​ ​ to​ ​ the​ ​ array.​ ​ Use​​ for-each-loop​ ​ to​ ​ print​ ​ out​ ​ the​​ info​ ​ of​​ each​ object ​ in​ ​ the​ ​ array​ ​ by​ ​ ​calling ​ the​​ say() ​ method.​​ Here​ ​ is​ ​ the​ ​ sample​​ output:​

Hello!​ ​My​ ​name​ ​is​ ​John​ ​Smith,​ ​and​ ​I'm​ ​in​ ​CPD​ ​program.

Hello!​ ​My​ ​name​ ​is​ ​Colin​ ​Thomas,​ ​and​ ​I'm​ ​in​ ​CPD​ ​program.

Hello!​ ​My​ ​name​ ​is​ ​Jordan​ ​Anastasiade,​ ​and​ ​I'm​ ​in​ ​T1034

Hello!​ ​My​ ​name​ ​is​ ​Wei​ ​Song,​ ​and​ ​I'm​ ​in​ ​T2099

Hello!​ ​My​ ​name​ ​is​ ​Jack​ ​Brown,​ ​and​ ​my​ ​title​ ​is​ ​Admin

Hello!​ ​My​ ​name​ ​is​ ​Paul​ ​Miller,​ ​and​ ​my​ ​title​ ​is​ ​Technical​ ​Support

IMPORTANT ​ REQUIREMENTS:​

* Each ​ class,​ ​ interface​ ​ should​ ​ be​​ public​ ​ and​ ​ have​ ​ its​ ​ own​ ​ source​ ​ ​file.
* One ​ (​ or ​ more)​ ​ package(s)​ ​ should​ ​ be​ ​ used​ ​ to​ ​ organize​ ​ ​your ​ ​project
* All ​ class​ ​ fields​ ​ must​ ​ be​​ private.​ ​ Getters​ ​ and​ ​ setters,​ ​ toString,​ ​ equal,​ ​ ​hashCode methods ​ must​ ​ be​ ​ implemented.​
* Write ​ the​ ​ documentation​ ​ properly​ ​ based​ ​ on​ ​ javadoc​ ​ requirements.​

@​ ​Wei​ ​Song,​ ​Jordan​ ​Anastasiade