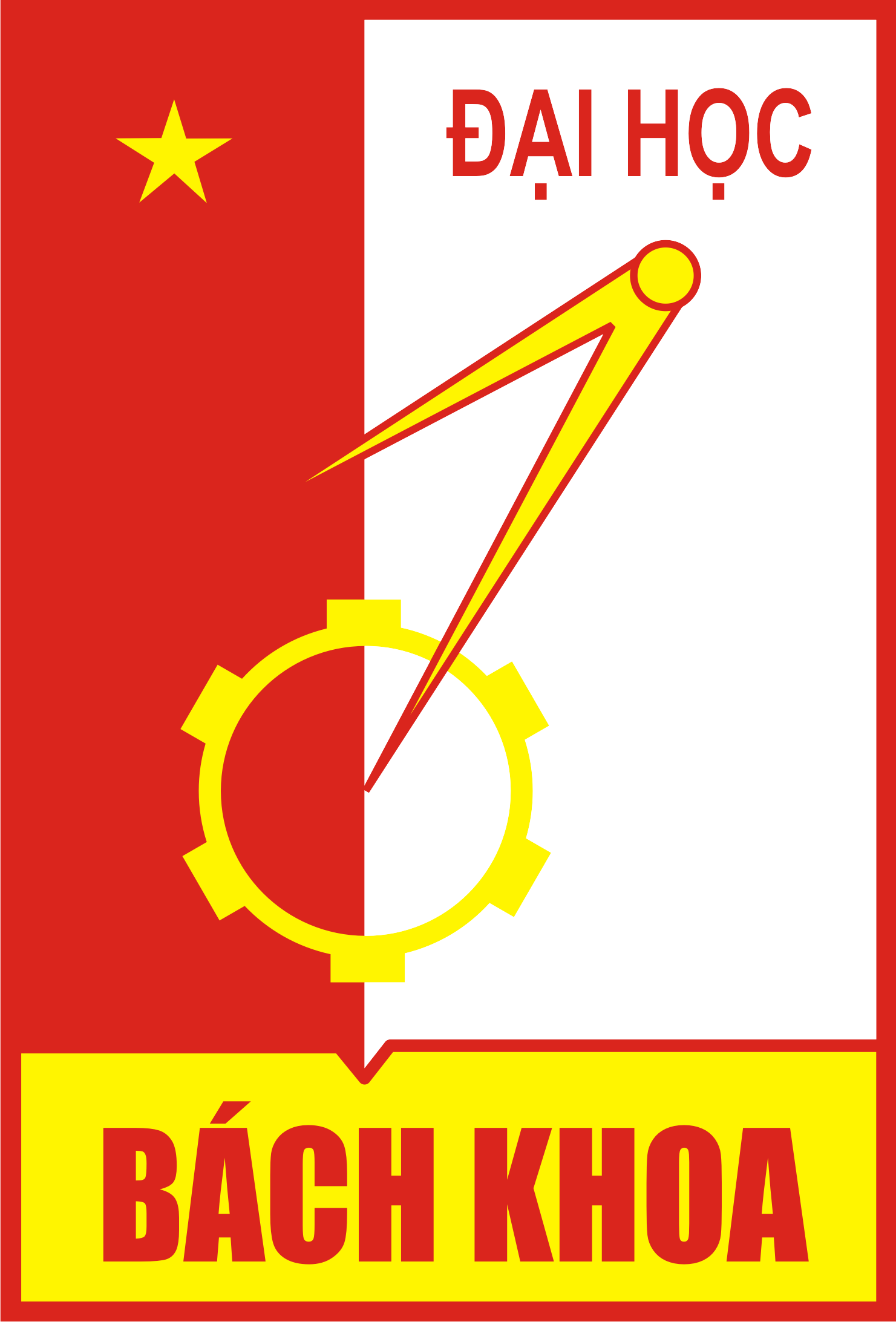
Hanoi University of Science and Technology  
School of Information and Communication Technology



**Mini-project Report**

**Subject: Object-Oriented Language and Theory (Java)**

***Topic: Semiconductor Visualization***

**Instructor: Nguyen Thi Thu Trang**

**Group 13**

|  |  |
| --- | --- |
| **Name** | **Student ID.** |
| **Trịnh Thu Hải** | **20184255** |
| **Nguyễn Thị Minh Châu** | **20184238** |

*Hanoi, 6/2020*

Table of Contents

[**1. Mini-project Description**](#_heading=h.gjdgxs) **3**

[1.1 Topic](#_heading=h.30j0zll) 3

[1.2 Detailed requirement](#_heading=h.1fob9te) 3

[*Doping*](#_heading=h.3znysh7) *3*

[*Temperature*](#_heading=h.2et92p0) *3*

[*External voltage*](#_heading=h.tyjcwt) *3*

[1.3 Use-case diagram](#_heading=h.i1zq4g4x011j) 4

[**2. Design Ideas**](#_heading=h.3dy6vkm) **5**

[2.1 Basic ideas](#_heading=h.9fhmrqhcyvkz) 5

[2.2 General class diagram](#_heading=h.gklzsevi6hkf) 6

[2.3 Detailed class Diagram](#_heading=h.o4wflq956p8q) 7

[2.3.1 Package: element](#_heading=h.c1cnymhkl8wa) 7

[2.3.2 Package: environment](#_heading=h.26l0am80gchp) 9

[2.3.3 Package: utils](#_heading=h.a1oa01gbhvia) 11

[2.3.4 Package: application](#_heading=h.xgmng5t7jb4l) 12

[**3. Assignment of members**](#_heading=h.1t3h5sf) **13**

[3.1 General assignment](#_heading=h.orey65e0thos) 13

[3.2 Detailed assignment](#_heading=h.fdx4vrn3h2j) 13

# 1. Mini-project Description

## 1.1 Topic

An application to (visually) demonstrate how semiconductors work, using object-oriented programming methodology.

## 1.2 Detailed requirement

In order to explain how semiconductors conduct or insulate electricity under different conditions, the app must allow users to observe visually the flow of charge carriers (electrons), inside a semiconductor crystal.

The electric flow might change in velocity (the average speed of charge carriers), or volume (the number of charge carriers flowing), under the effects of the following factors:

### Doping

A pure semiconductor has no free electron and no mobile holes (all electrons are bounded inside a valence bond), so in normal conditions, virtually no flow is detected

A semiconductor doped with P-typed atoms has 1 free electron for each P-type atom, these free electrons can flow with external voltage

A semiconductor doped with N-typed atoms has 1 mobile hole (an absent electron) for each N-type atom, these mobile holes can flow with external voltage

Dope level: semiconductors of type P and N can be doped in different levels. Higher dope level indicates more impure atoms in the material.

### Temperature

Increasing temperature can give energy to the bounded electrons in the semiconductor, thus, turning them into free electrons and create mobile holes in the position of the valence bonds where the bounded electrons used to be. This reaction increases the volume of the flow.

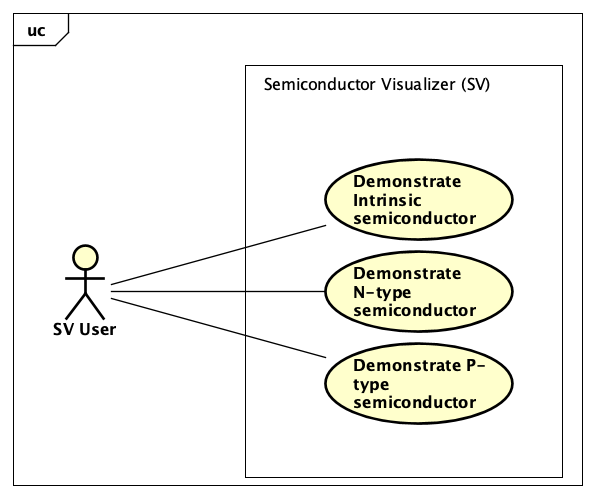
Decreasing temperature, in reverse, can decrease the volume of the flow

Temperature also affects movement of the free electrons. The higher the temperature, the more chaotically the electrons move.

### External voltage

As external voltage increases, the velocity of the flow increases and vice versa.

## 1.3 Use-case diagram



***Figure 1:*** *Use-case diagram of the application*

The program has 3 use-cases:

* Demonstrate the working principles of intrinsic semiconductor
* Demonstrate the working principles of P-type semiconductor
* Demonstrate the working principles of N-type semiconductor

In each use-case, users can

* vary conditions for visualising the material in different situations. For example: external voltage and temperature applying on the material.
* change doping level (ie. lightly doped or heavily doped),
* start and stop the simulation

# 2. Design Ideas

## 2.1 Basic ideas

The application is built to visualize the basic operations of 3 types of semiconductors:

* Intrinsic type: purely consists of Silicon atoms
* P-type: consists of Silicon and Phosphorus atoms
* N-type: consists of Silicon and Aluminum atoms

Each type of the semiconductor will be constructed in the form of a 5x6 crystal, each cell will either be a Silicon, a Phosphorus, or an Aluminum atom.

The atom of each element will have the following components, called charge:

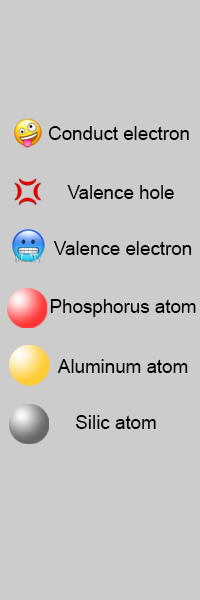
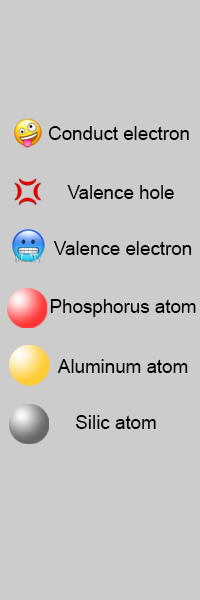
* conduct electron: electron in conduction band
* valence electron: electron in valence band
* valence hole: a gap in valence band

Number of each type of charge in each type of element is shown in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Atom of element | #. Hole | #. Valence electron | #. Conduct electron |
| Silicon | 0 | 4 | 0 |
| Phosphorus | 0 | 4 | 1 |
| Aluminum | 1 | 3 | 0 |

***Table 1:*** *Number of types of charge in each atom of different elements.*

The representation of holes, valence electrons, conduct electrons and different nuclei (due to various types of elements) are chosen differently for easily identifying the components of each atom being shown on the visualizer.



***Figure 2:*** *Representation of different components of atom in different elements*

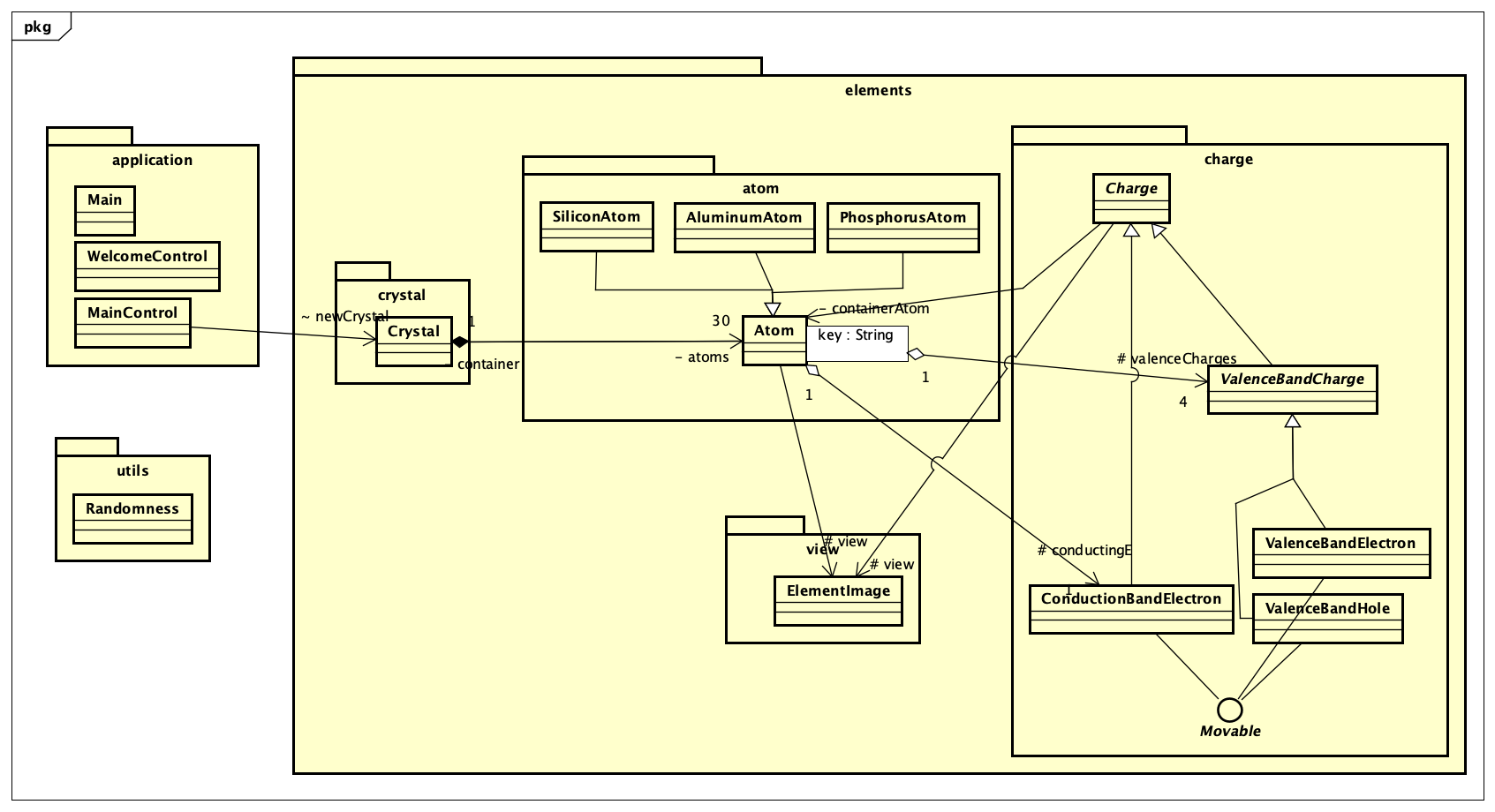
Users must choose a type of semiconductor before getting into the application. The program will automatically choose dope type of Light. However, these parameters can be changed while using the app.

The visualization will start after receiving a start request from the user. At this moment, users can vary applied temperature and external voltage onto the crystal to see the change in movement of particles inside the semiconductor in different situations. Users can also set the temperature and external voltage condition before starting the simulation.

After having observed the working principle of the semiconductor, users can choose to stop the visualization or exit the program.

Users can also see the instruction and information of the application when using the program.

## 2.2 General class diagram



***Figure 3:*** *General diagram of the application*

The application consists of 3 main packages

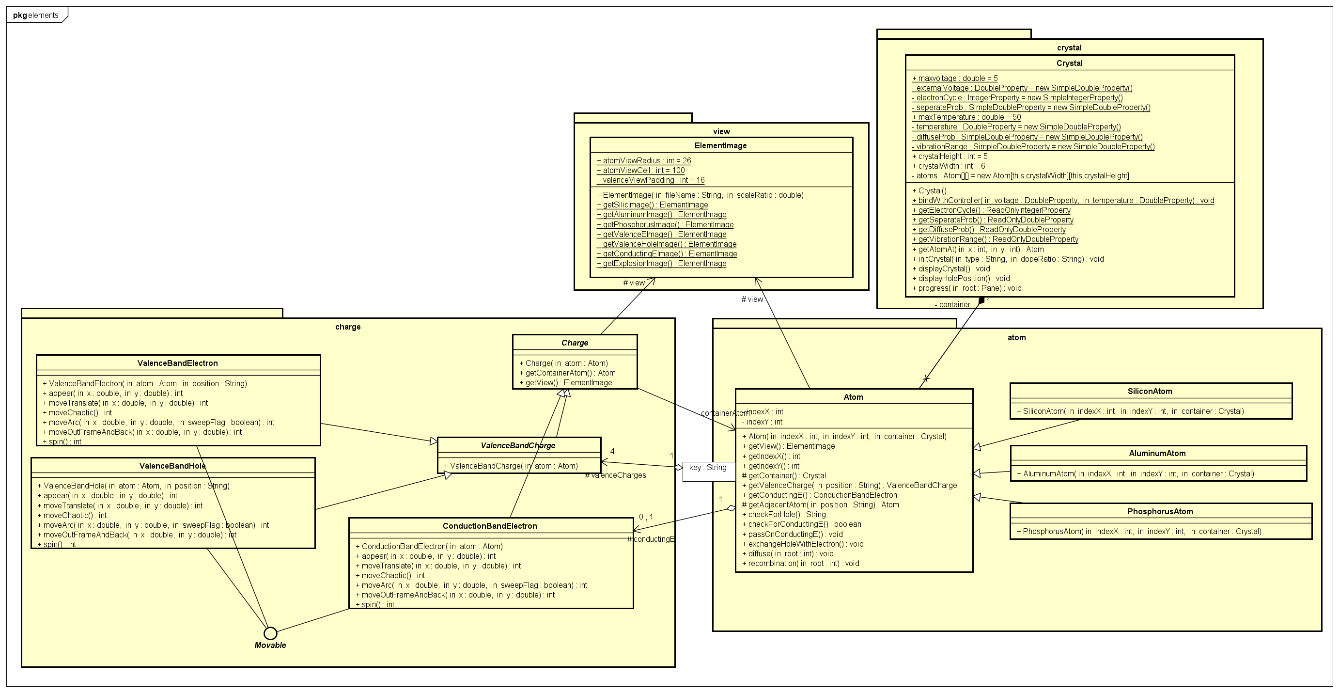
**elements:** contains 4 sub-packages: atom, charge, crystal, view. These packages contain interface and classes used for defining each elements’ attributes and method

**utils:**contains additional methods for constructing the elements’ and semiconductor crystal.

**application:** contains classes for demonstrating the application to users.

## 2.3 Detailed class Diagram

#### 2.3.1 Package: element



***Figure 4:*** *Class diagram of package elements*

**Package: elements:** consists of 4 subpackages

***elements.atoms***: Consists of 4 class:

**Atom:** Parent class for 3 child class: **AluminumAtom, PhosphorusAtom** and **SiliconAtom**

• Attributes:

* indexX: horizontal position of atom in the crystal
* indexY: vertical position of aton in the crystal

• Constructors:

* Atom(int indexX, int indexY, Crystal container): set up an atom at row indexX, column indexY of crystal container.

• Methods:

* getView(): return image representation of the atom
* getIndexX(): return horizontal position (row) of the atom
* getIndexY(): return vertical position (column) of the atom

# getContainer(): return Crystal container that the atom is in.

* getValenceCharge(String position): return valence charge of the atom at specified position (if exist)
* getConductingE(): return conducting electron of the atom (if exist)

# getAdjacentAtom(String position): return nearby atom at specified position

* checkForHole(): return a string representing the position of hole in the atom. If there is not anyhole, return “none”
* checkForConductingE(): return true if there is a conducting electron in the atom
* passOnConductingE(): pass conducting electron to nearby atom
* exchangeHoleWithElectron(): exchange hole with electron if reaching conditions
* diffuse(): change valence electron into hole and conducting electron
* recombination(): combine conducting electron and hole into valence electron.

***elements.charge:*** consists of 5 classes:

**Charge:** parent class of 2 child classes: **ConductionBandElectron** and **ValenceBandCharge**

**ValenceBandCharge:** parent of 2 child classes: **ValenceBandElectron** and **ValenceBandHole**

**ConductionBandCharge**, **ConductionBandElectron** and **ConductionBandHole** inherits ***interface Movable***, with the following methods:

* + appear(): appear on the visualizing pane
  + moveTranslate(): moving translational on visualizing pane
  + moveChaotic(): vibrate
  + moveArc(): moving in an arc
  + moveOutOfFrameAndBack(): when the particle reaches the right end of the crystal, it will continue moving and return to the left end.
  + spin(): spin

ValenceBandCharge and ConductionBandElectrons aggregate with Atoms, since atoms do not have to contain all types of charges and still exist if there is no charge available.

Each Atom can either have 0 or 1 ConductionBandElectron. Each Atom have 4 ValenceBandCharge.

***elements.crystal:*** consists of 1 class

**Crystal**

• Attributes:

* crystalHeight: height of the crystal. Initial value: 5
* crystalWidth: width of the crystal. Initial value: 6

• Methods:

* getAtomAt(int x, int y): return atom at position (x,y)
* initCrystal(String type, String dopeRatio): initialize a crystal of type type and dope level dopeRatio.
* progress(Pane root): start performing operations of particles in the crystal displayed on pane root

Crystal and Atom have Composition relationship: if a crystal is destroyed, all atoms inside the crystal will also be destroyed.

***elements.view:*** consists of 1 class, use for displaying elements on GUI

**ElementImage**

• Attributes:

+ atomViewRadius: setting the radius of the central nuclei. Initial value: 26

+ atomViewCell: setting the distance between 2 center point of 2 adjacent atoms. Initial value: 100

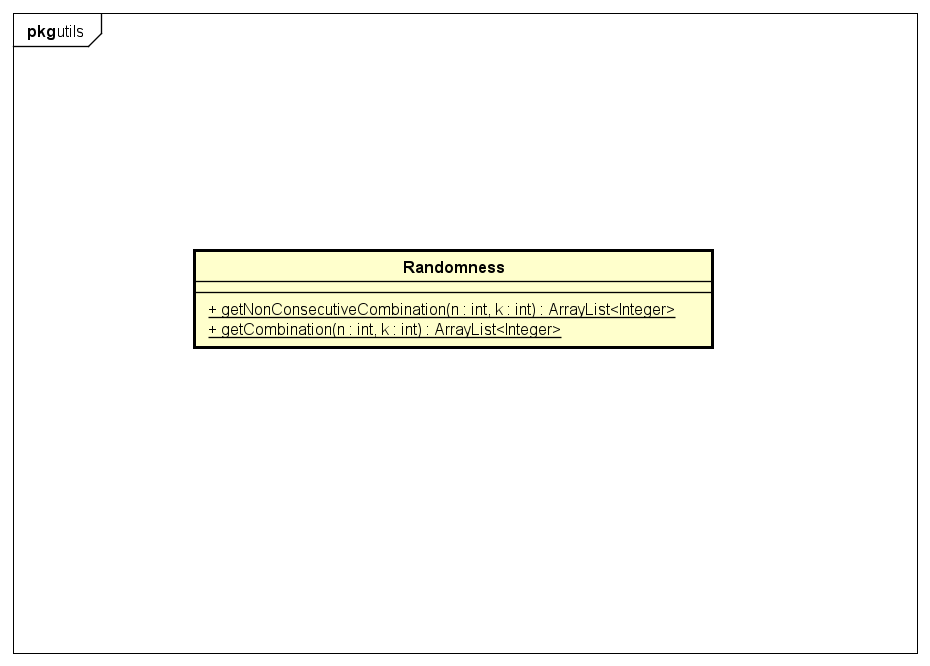
+ valenceViewPadding: setting the gap between the central nuclei and the valance band elements. Initial value: 16

• Methods:

* ElementImage(String fileName, double scaleRatio): load image of element and scale it with ratio scaleRatio
* get() methods: return image of element.

Charge and Atom have Composition relationship: if an atom is destroyed, all charge inside the atom will also be destroyed.

#### 2.3.2 Package: utils



***Figure 6:*** *Class diagram of package utils*

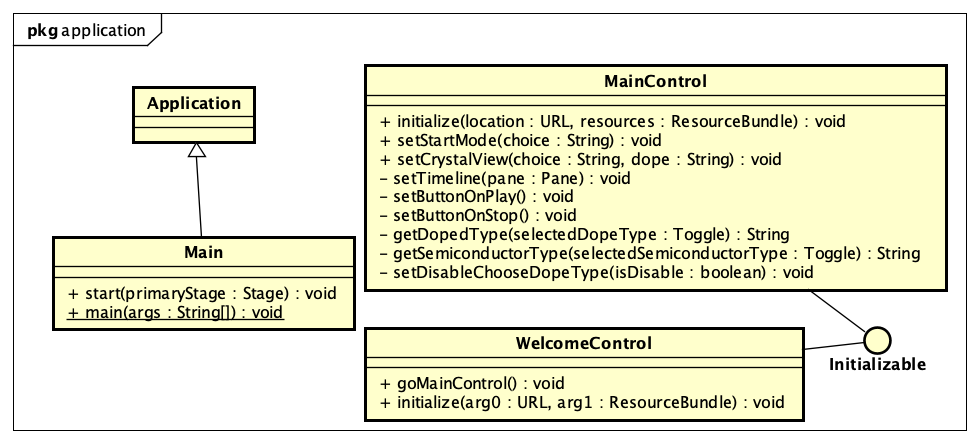
**Package: utils:** consists of 1 class

**Randomness:** additional methods for initializing the crystal

• Methods:

* getNonConsecutiveCombination(): get a random combination of non-consecutive k integers in the range [0, n-1], used in class **Crystal** to initialize doping position
* getCombination(): get a random combination of k integers in the range [0, n-1]

#### 2.3.3 Package: application



***Figure 7:*** *Class diagram of package application*

**Package: application:**  consists of 3 classes:

**Main:** derived from abstract class Application, override start() method, use for starting application.

**WelcomeControl:** inherits interface Initializable, overrides initialize() method for setting up components on welcome screen. WelcomeControl class has one method

* goMainControl(): for getting to the main window of the application after users have chosen visualize mode.

**MainControl:** inherits interface Initializable, overrides initialize() method for setting up components on main GUI. MainControl class has some additional methods for helping getting/setting components of the GUI

* setStartMode(String choice): for setting start mode of the application, based on the choice of user at the welcome screen.
* setCrystalView(String choice, String dope) for setting initial simulation of semiconductor crystal on the main pane, take 2 parameters: choice for type of the material (P, N, instrinse) and dope (light and heavy dope level).
* setTimeline(Pane pane): for setting up timeline for animation of elements’ inside the crystal on the main pane in different situations.
* setButtonOnPlay(): for reset buttons when users perform operations on buttons when the visualization is being conducted.
* setButtonOnStop(): for reset buttons when users perform operations on buttons when the visualization is not being conducted.
* getDopedType(Toggle selectedDopeType): get selected dope level (from ToogleGroup) for setting up the crystal. This method is used for the situation when users change simulation mode after getting to the application.
* getDopedType(Toggle selectedDopeType): get selected dope level (from ToogleGroup) for setting up the crystal. This method is used for the situation when users change simulation mode after getting to the application.

# 3. Assignment of members

## 3.1 General assignment

|  |  |
| --- | --- |
| Name | Assignment |
| Trịnh Thu Hải | Design ideas  Design class diagram & use-case diagram  Implement backend codes  Test backend  Suggest modifications to frontend  Integrate frontend and backend code |
| Nguyễn Thị Minh Châu | Design ideas  Design class diagram & use-case diagram  Implement frontend codes & design GUI  Test frontend and GUI  Suggest modifications to backend  Final test  Write documents, preparing video & presentation slides. |

## 3.2 Detailed assignment

|  |  |
| --- | --- |
| Name | Assignment |
| Trịnh Thu Hải | Package: elements.atom  • Class   * Atom * AluminumAtom * PhosphorusAtom * SiliconAtom   Package: elements.charge  • Class   * Charge * ConductionBandElectron (all methods not inherited from Movable) * ValanceBandCharge * ValenceBandElectron (all methods not inherited from Movable) * ValenceBandHole (all methods not inherited from Movable)   Package: elements.crystal  • Class   * Crystal   Package: elements.view  • Class   * ElementImage   Package: environment  • Class   * Environment   Package: utils  • Class   * Randomness |
| Nguyễn Thị Minh Châu | Package: images  Package: application  • Class:   * Main * MainControl * WelcomeControl   • UI Design   * AboutWindow * HowToUseWindow * MainWindow\_1 * WelcomeWindow   Package: elements.charge  • Interface: Movable   * appear(); * moveArc(); * moveChaotic(); * moveTranslate(); * spin();   • Class:   * ConductionBandElectron (the inherited methods from Movable) * ValenceBandElectron (the inherited methods from Movable) * ValenceBandHole (the inherited methods from Movable) |
|  | \*\* no specifications means implementing all methods in class |