Lab 2: Java Inheritance

Instruction

1. Click the provided link on CourseVille to create your own repository.
2. Open Eclipse and then "File > new > Java Project" and set project name in this format **2110215_Lab2_2020_2_{ID}_{FIRSTNAME}**
    - Example: **2110215_Lab2_2020_2_6331234521_Samatcha**.
3. Initialize git in your project directory
    - Add .gitignore and set up your git.
    - Create your remote repository from a given link.
    - Commit and push initial codes to your GitHub repository.
4. Implement all the classes and methods following the details given in the problem statement file which you can download from CourseVille.
    - You should create commits with meaningful messages when you finish each part of your program.
    - Don't wait until you finish all features to create a commit.
5. Test your codes with the provided JUnit test cases, they are inside package **test.grader**
    - If you want to create your own test cases, please put them inside package **test.student**
    - Aside from passing all test cases, your program must be able to run properly without any runtime errors.
6. After finishing the program, create a UML diagram using **ObjectAid** and put the result image (UML.png) at the root of your project folder.
7. Export your project into a jar file called **Lab2_2020_2_{ID}** and place it at the root directory of your project. Make sure you export all your source (.java) files. You can open the jar file with any zip software to check it.
    - Example: **Lab2_2020_2_6331234521.jar**
8. Push all other commits to your GitHub repository.

1. Problem Statement

Space is one of the places that we want to go to in our childhood. However, this kind of dream costs us lots of money. Being just a computer geek, we are going to create our own virtual solar system.

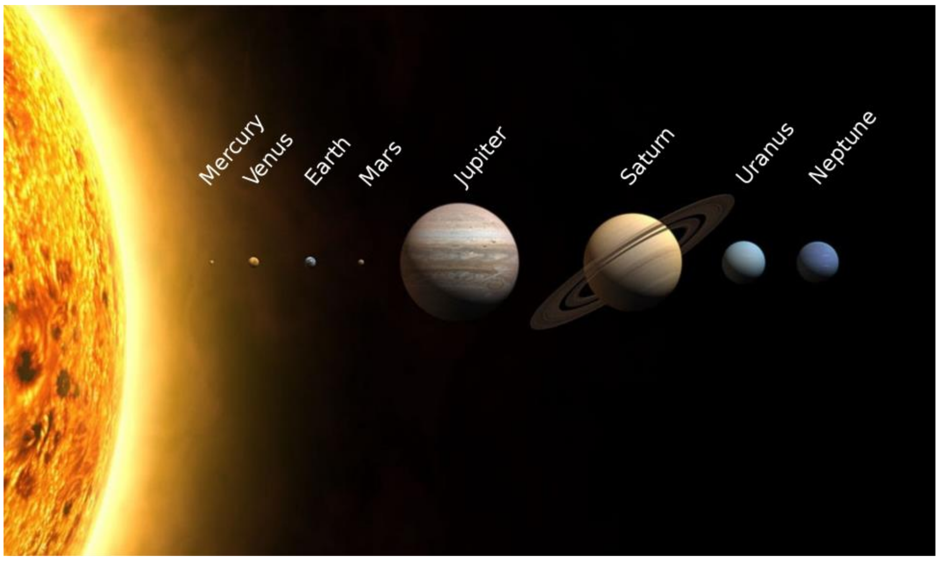

Figure 1: Solar System

(https://upload.wikimedia.org/wikipedia/commons/thumb/c/cb/Planets2013.svg/2000px-Planets2013.svg.png)

Our Solar System consists of many planets orbiting around the Sun. For simplicity, we will choose only two planets: **Earth** and **Saturn**. In addition, the orbit will be simple too because the planet can only orbit around these **4 positions** (shown in **Figure 2**) that far away from the Sun defined by orbit radius (each planet may have different orbit radius).

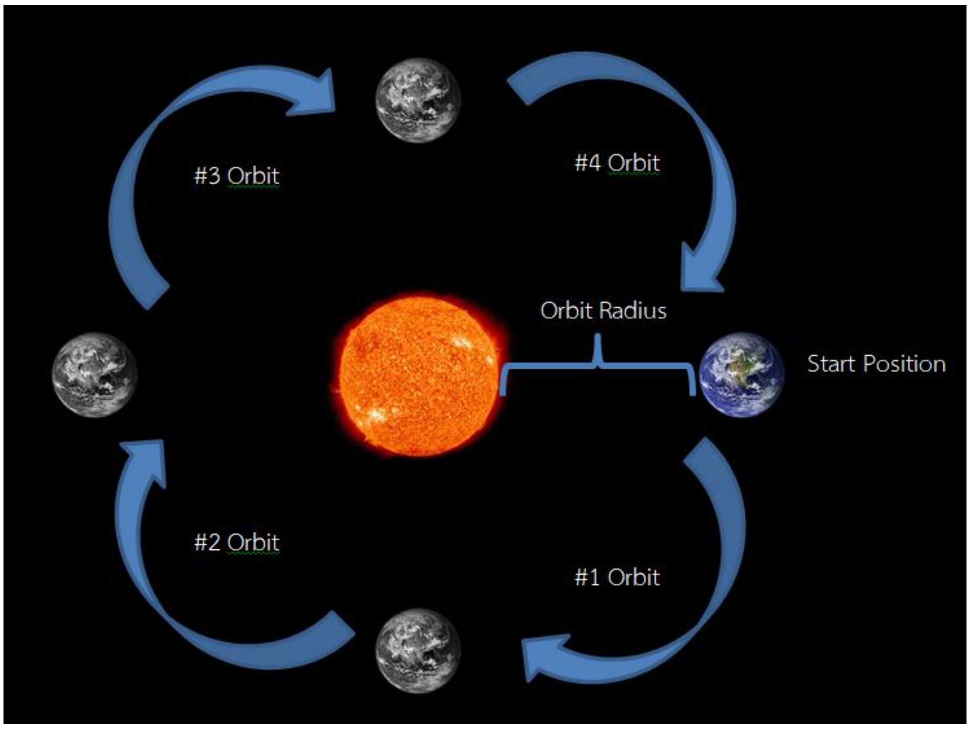

Figure 2: Orbit System

However, the Saturn class is still incomplete. So it's your task to help fulfill our childhood dream.

For better understanding of this, there is a **SolarSystem** class in package **solarsystem** which will provide the real usage of the planets. The document about how to use it is available as **SolarSystem.pdf**.

2. Implementation Detail

The class-package is summarized below

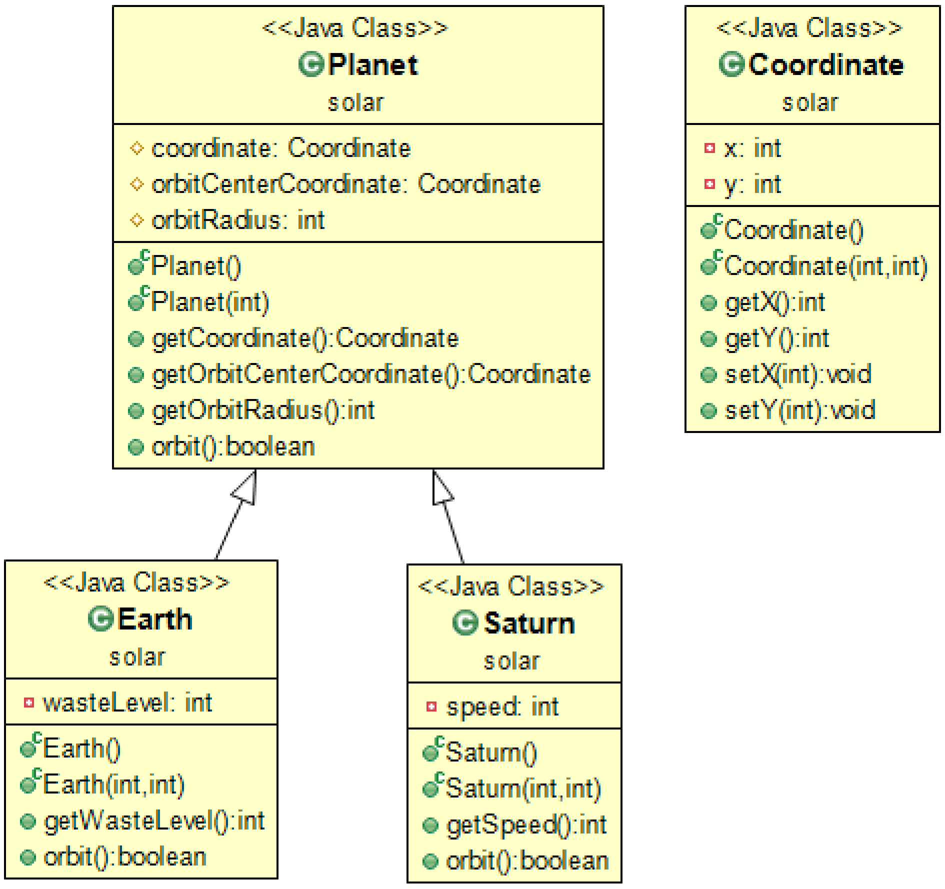

Figure 4 Class diagram

You must write java class Saturn from scratch using UML diagram specified above.

* In the following class description, only details of IMPORTANT fields and methods are given. *

4.1 Package solar

Class *Coordinate* /*Fill Code*/

Field

- **- int x** - The position of axis X.
- **- int y** - The position of axis Y.

Constructor

- **+ Coordinate()** - Set related fields with default values (All field will be set to 0).
- **+ Coordinate(int x, int y)** - Set related fields with the given parameters.

Method

- **+ int getX()** - Return the position of axis X.
- **+ int getY()** - Return the position of axis Y.
- **+ void setX(int x)** - Set the position of axis X equals to x.
- **+ void setY(int y)** - Set the position of axis Y equals to y.

Class *Planet* /*Fill Code*/

Field

- **# Coordinate coordinate** - The current coordinate of this planet.
- **# Coordinate orbitCenterCoordinate** - The orbit center coordinate used for orbiting around.
- **# int orbitRadius** - The orbit radius of this planet (the distance from the planet's coordinate to the orbit center coordinate).

Constructor

- **+ Planet()** - Set related fields with default values (coordinate will be set to (1,0) which will correspond to orbitRadius with value 1).
- **+ Planet(int orbitRadius)** - Set related fields with the given parameters; note that the orbitRadius must be at least 1.

Method

- **+ Coordinate getCoordinate()** - Return current coordinate.
- **+ Coordinate getOrbitCenterCoordinate()** - Return orbit center coordinate.
- **+ int getOrbitRadius()** - Return the orbit radius.
- **+ boolean orbit()** - Return true if the orbit can occur. The planet will orbit only to 4 positions (far from center with orbit radius) clockwisely around the center. Otherwise, return false.

Class *Earth* /*Fill Code*/

Field

- **- int wasteLevel** - The Earth's waste level.

Constructor

- **+ Earth()** - Set related fields with default values. The waste level will be set to zero.
- **+ Earth(int orbitRadius, int wasteLevel)** - Set related fields with the given parameters; note that wasteLevel can't be below 0.

Method

- **+ int getWasteLevel()** - Return Earth's waste level.
- **+ boolean orbit()** - Return true if the orbit can occur. For Earth, the orbit can occur only when the Earth's waste level is less than or equal 5. Otherwise, return false.

Class *Saturn* /*Fill Code*/

Field

- **- int speed** - Number of times that the Saturn will orbit when the orbit command is executed.

Constructor

- **+ Saturn()** - Set related fields with default values. The speed will be set to zero.
- **+ Saturn(int orbitRadius, int speed)** - Set related fields with the given parameters; note that speed can't be below zero.

Method

- **+ int getSpeed()** - Return Saturn's speed.
- **+ boolean orbit()** - Return true if the orbit can occur. For Saturn, the orbit can occur only when the speed is more than zero (the number of orbit times will

equal to the value of speed eg. speed is 2 means orbits 2 times). Otherwise, return false.

Class *Application* /*Fill Code*/

Field

- - ArrayList<Planet> planets - An ArrayList of Planet.

Constructor

- + String printPlanet(Planet planet) - Return planet's specific attribute in the format specified below. If the planet isn't Earth or Saturn return empty String.
    - {PLANET_NAME}'s {SPECIFIC_ATTRIBUTE} is {SPECIFIC_ATTRIBUTE_VALUE}
        - eg. "Earth's Waste Level is 4 "
        - eg2. "Saturn's Speed is 2 "

Method

- + void main() - Create Earth with orbit radius equals 1 and waste level equals 4 and Saturn with orbit radius equals 2 and speed equals 2. Then, add both planets to the ArrayList of Planet. Finally, use printPlanet() to show the specific attribute of each planet (**Note that: toString() method for each planet is forbidden**).

4.2 Package SolarSystem

Class solarsystem

Field

- /*Fill Code*/ + ArrayList<Planet> planets - *An ArrayList of planet*
- + int X_RANGE - range x-axis in solar system.
- + int Y_RANGE - range y-axis in solar system.

Method

- + void main() - show command menu
- + void addPlanet(Planet planet) - /*Fill Code*/ Add planet if the planet does not existed and use printMap() to show solar system.

- + boolean doesPlanetExist(Planet planet) - /*Fill Code*/ If planet does existed "print(planet.getClass().toString().split(" ")[1] + " exists !!!")" and return true else return false.
- + void printMap() - Show solar system.

Score Criteria (13) (will be rounded to 5)

ApplicationTest	3
testPrintEarth	1
testPrintOtherPlanet	1
testPrintSaturn	1
EarthTest	5
testConstructor1	1
testConstructor2	1
testOrbit	2
testgetWasteLevel	1
SaturnTest	5
testConstructor1	1
testConstructor2	1
testGetSpeed	1
testOrbit	2