Intermediate Software Development – MiniProject 1

Project Description:

The goal of this project is to create object model for Solar System. An abstract super class called Planet and subclass called MilkyWayPlanet, with default constructor (no-argument) and full-argument overloaded constructors should be provided. Include an Interface class named Relatable which has two methods isMassSmaller() and isDiameterGreater() which compares the corresponding mass and diameter of any two planets. Ultimately, a driver class should be added in which instances are created for each planet, collects all planet objects into a Planet[ ] array named solarSystem, implement the polymorphic behavior, compare the mass of any two planets of our choice, compare the diameter of any two planets of our choice.

Installation, Compile and Run Time requirements:

1. NetBeans IDE 7.2.1
2. Java 1.7.0_11, Java Hotspot(TM) 64-Bit Server VM 23.6-b04
3. Windows 7 version 6.1 running on amd64

Insights, Expected results, and Challenges:

Insights:
This project helped me get hands on experience on the concepts of Inheritance, Polymorphism and Abstraction.
Inheritance was demonstrated by the subclass, which inherited the functionality from the super class and the interface.

The ‘Planet’ class demonstrated the characteristics of the ‘MilkyWayPlanet’ when it is instantiated with the subclass type. Thus demonstrating the concept of Polymorphism.

The concept of ‘Abstraction’ was demonstrated as follows:
The super class ‘Planet’ was made ‘abstract’, which means that the class cannot be instantiated and it is incomplete. However, the functionality of this class will be used by its subclass. I have learnt that by designing the abstract super class in a way that all the common fields are defined in it, it becomes easier to share the characteristics between the subclasses.

The packages used in this project are:
- domain
- driver

domain package consists of the following classes:
1. Planet
2. MilkyWayPlanet
3. Relatable

1**. Planet Superclass:**

This class is an abstract super class. This is the parent class and it performs the following operations:

I. Import the required packages.
II. Declaration of variables.
III. Implementing the interface from the package Relatable.
IV. Providing No-arguments constructor with default values.
V. Providing Full-arguments constructor.
VI. Providing accessors and mutators for the variables declared.
VII. Providing toString() method.

2. **MilkWayPlanet Subclass:**

This class is a subclass which does the following operations:

I. Import the required package
II. Inherit the parent class i.e. the abstract super class 'Planet'
III. Create an instance of each planet.

3. **Relatable Interface class:**

This is an interface which has the following methods:

I. boolean isMassSmaller(Object other) :

Logic:

- □ Initialize a boolean variable result to false.
- □ Upcast the Object other to Planet super class
- □ Compare the mass by retrieving the value from respective planets using getmass() method.
- □ Change the value of boolean variable result if condition is true
- □ Return the result value.

II. boolean isDiameterGreater(Object other):

Logic:

- □ Initialize a boolean variable result to false.
- □ Upcast the Object other to Planet super class
- □ Compare the diameter by retrieving the value from respective planets using getmass() method.
- □ Change the value of boolean variable result if condition is true
- □ Return the result value.

driver package consists of the following class :

1. **Main Driver:**

This is the main class from where the project execution starts and performs the following operations:

I. Planet's instance of MilkyWayPlanet is created.
II. Planet's data will be displayed.
III. All the Planet's objects are collected into an array of Planet called solarSystem.
IV. Implements polymorphism behavior where it displays Planet's data from solarSystem variable (Type Planet) using toString() method.
V. Invokes the isMassSmaller() comparison method for Mercury and Jupiter to compare the mass between two planets and displays which smaller. Invokes the

isDiameterGreater() comparison method for Mercury and Jupiter to compare the diameter between two planets and displays which is larger planet.

Expected Results:

Expected Results	Test Result
1. All nine planets' data (created as instance of MilkyWayPlanet) should be displayed. [Not Mandatory as per the requirement]	Pass
2. All nine planets' data should use toString() method there by implementing polymorphism behavior(created by collecting all nine planets' object into an array of Planet called solarSystem).	Pass
3. Should display the result of comparison of any two planet's mass out of nine planets.	Pass
4. Should display the result of comparison of any two planet's diameter out of nine planets.	Pass

Challenges Faced:

While coding the project, there were no major challenges as such. But did encounter minor hiccups while creating the test script that will write the output to a text file. Initially when I created the script file and executed, it created an empty text file. I tried several options to resolve it. Finally, I understood the problem was with jar file. So, I cleaned the project and build it which created a new jar file. Then when I executed the script file the output was successfully generated into text file as expected.
Also, while populating data for instance fields of all planets' I noticed that the value of mass, revolution period etc varies from planet to planet in unit. For example Mercury revolution period was in days unit but Earth was in hours unit. Then I decided to use consistent unit for all the instance fields so that the result of mass and diameter comparisons would be accurate between any two planets.

Screenshots:

1. **Planet Superclass:**

Snapshot 1-1:

Instance fields of planets with default(no arg) and overloaded(full arg) constructors of Planet class.

```
public abstract class Planet implements Relatable {

    protected String planetName; //Name of the galaxy planets.
    protected double mass; //Mass of planets in 10 power 24 kg unit.
    protected int diameter; // Diameter of planets in Km(Kilometer) Unit.
    protected double escapeVelocity; //Escape velocity in Km/s(Kilometer/sec) Unit.
    protected double revolutionPeriod; //Revolution period of planets in Hours Unit.
    protected int meanSurfaceTemperature;//Surface Temperature of planets in K(Kelvin Scale) Unit.

    public Planet() {
        planetName = "Null";
        mass = 0.0;
        diameter = 0;
        escapeVelocity = 0.0;
        revolutionPeriod = 0.0;
        meanSurfaceTemperature = 0;
    }

    public Planet(String planetName, double mass, int diameter, double escapeVelocity, double revolutionPeriod, int meanSurfaceTemperature) {
        this.planetName = planetName;
        this.mass = mass;
        this.diameter = diameter;
        this.escapeVelocity = escapeVelocity;
        this.revolutionPeriod = revolutionPeriod;
        this.meanSurfaceTemperature = meanSurfaceTemperature;
    }
```

Snapshot 1-2:

```
    //Accessors and Mutators of instance fields
    public String getPlanetName() {
        return planetName;
    }

    public void setPlanetName(String planetName) {
        this.planetName = planetName;
    }

    public double getMass() {
        return mass;
    }

    public void setMass(double mass) {
        this.mass = mass;
    }

    public double getDiameter() {
        return diameter;
    }

    public void setDiameter(int diameter) {
        this.diameter = diameter;
    }

    public double getEscapeVelocity() {
        return escapeVelocity;
    }
```

Accessors and Mutators of instance fields declared in Planet class.

Snapshot 1-3:

```java
    public void setEscapeVelocity(int escapeVelocity) {
        this.escapeVelocity = escapeVelocity;
    }

    public double getRevolutionPeriod() {
        return revolutionPeriod;
    }

    public void setRevolutionPeriod(double revolutionPeriod) {
        this.revolutionPeriod = revolutionPeriod;
    }

    public int getMeanSurfaceTemperature() {
        return meanSurfaceTemperature;
    }

    public void setMeanSurfaceTemperature(int meanSurfaceTemperature) {
        this.meanSurfaceTemperature = meanSurfaceTemperature;
    }

    @Override
    public String toString() {
        return "Planet{" + "Planet Name = " + planetName +
                ", mass = " + mass + "x" + Math.pow(10, 24) + "Kg" +
                ", diameter = " + diameter + "km" +
                ", escapeVelocity = " + escapeVelocity + "Km/s" +
                ", revolutionPeriod = " + revolutionPeriod + "Hours" +
                ", meanSurfaceTemperature = " + meanSurfaceTemperature + "K" + "}";
    }
```

Accessors and Mutators of Planet superclass with toString() method to format planets data.

2. **MilkyWayPlanet subclass :**

Snapshot 2-1:

```java
 */
package domain;

/**
 *
 * @author Meghashree
 */
public class MilkyWayPlanet extends Planet {

    //No argument Constructor
    public MilkyWayPlanet() {
        super();
    }

    //full argument constructor
    public MilkyWayPlanet(String planetName, double mass, int diameter, double escapeVelocity, double revolutionPeriod, int meanSurfaceTemperature) {
        super(planetName, mass, diameter, escapeVelocity, revolutionPeriod, meanSurfaceTemperature);

    }

    @Override
    public String toString() {
        return "Milkywayplanets{" + super.toString() + "}";
    }

    //Interface methods used
    //to compare mass of two planets
```

MilkyWayPlanet subclass with default(no arg) and a full arg, overloaded constructors and toString() method.

Snapshot 2-2:

```
    //Interface methods used
    //to compare mass of two planets
    @Override
    public boolean isMassSmaller(Object other) {
        boolean result = false;
        Planet p = null;
        if (other instanceof Planet) {
            p = (Planet) other;
        }
        if (p != null) {
            if (this.getMass() < p.getMass()) {
                result = true;
            }
        }
        return result;
    }
    //Interface methods used
    //to compare diameter of two planets
    @Override
    public boolean isDiameterGreater(Object other) {
        boolean result = false;
        Planet p = null;
        if (other instanceof Planet) {
            p = (Planet) other;
        }
        if (p != null) {
            if (this.getDiameter() > p.getDiameter()) {
                result = true;
            }
        }
        return result;
    }
```

Interface Methods isMassSmaller(Object other) and isDiameterGreater(Object other) used in Planet Class to compare mass and diameter of any two planets

3. **Relatable Interface class:**

Snapshot 3-1

```
Relatable.java
ource History
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package domain;

/**
 *
 * @author MeghaVinay
 */
public interface Relatable {

    boolean isMassSmaller(Object other);

    boolean isDiameterGreater(Object other);
}
```

Interface methods isMassSmaller(Object other) and isDiameterGreater(Object other) of Relatable interface class. These methods are further used in Planet class

4. **MainDriver driver class:**

Note: Currently the System.out.println("Planets……); shown in Snapshot 4-1 is commented in the code as it wasn't the part of the requirement.

Snapshot 4-1:

```
MainDriver.java
public static void main(String[] args) {
        // TODO code application logic here
        //Instance of each MilkyWayPlanet class
        MilkyWayPlanet Mercury = new MilkyWayPlanet("Mercury", 33.02, 4878, 4.25, 1407.6, 623);
        MilkyWayPlanet Venus = new MilkyWayPlanet("Venus", 4.87, 12104, 10.4, 5832.24, 750);
        MilkyWayPlanet Earth = new MilkyWayPlanet("Earth", 5.976, 12756, 11.2, 23.93, 293);
        MilkyWayPlanet Mars = new MilkyWayPlanet("Mars", 64.2, 6794, 3, 24, 293);// revolution period inhours
        MilkyWayPlanet Jupiter = new MilkyWayPlanet("Jupiter", 0.63, 142800, 60, 10, 163);
        MilkyWayPlanet Saturn = new MilkyWayPlanet("Saturn", 0.0569, 120000, 35.6, 10, 93);
        MilkyWayPlanet Uranus = new MilkyWayPlanet("Uranus", 0.87, 51120, 21.2, 17.2, 57);
        MilkyWayPlanet Neptune = new MilkyWayPlanet("Neptune", 0.0103, 49528, 23.6, 16.11, 57);
        MilkyWayPlanet Pluto = new MilkyWayPlanet("Pluto", 0.01, 2290, 1, 153.29, 50);

        System.out.println("Planets Data : \n"
                + Mercury + "\n"
                + Venus + "\n"
                + Earth + "\n"
                + Mars + "\n"
                + Jupiter + "\n"
                + Saturn + "\n"
                + Uranus + "\n"
                + Neptune + "\n"
                + Pluto + "\n");

        // Planet objects into an array of Planets called solarSystem
        // ie.Upcast to Planet abstract class.
        //(Polymorphism : overrides toString at subclass level(MilkyWayPlanet)
        //even though upcasted to superclass(Planets))
        Planet[] solarSystem = {Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto};
        System.out.println("Polymorphism(Upcasted to Planet abstract superclass) :");
```

Code to create instance of each MilkyWayPlanet and print the objects data at output

Snapshot 4-2:

```
MainDriver.java
        + Pluto + "\n");

// Planet objects into an array of Planets called solarSystem
// ie.Upcast to Planet abstract class.
//(Polymorphism : overrides toString at subclass level(MilkyWayPlanet)
//even though upcasted to superclass(Planets))
Planet[] solarSystem = {Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto};
System.out.println("Polymorphism(Upcasted to Planet abstract superclass) :");
for (Planet galaxy : solarSystem) {
    System.out.println(galaxy);
}

// Upcast to Object class...
//(Polymorphism : overrides toString at subclass level(MilkyWayPlanet)
//even though upcasted to Super-superclass(Objects))
Object[] objects = {Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto};
System.out.println("\nPolymorphism(Upcasted to Object super-superclass) :");
for (Object obj : objects) {
    System.out.println(obj);
}

//Comparing Mass of Mercury and Jupiter
if (Mercury.isMassSmaller(Jupiter)) {
    System.out.println("\nMercury Mass : " + Mercury.getMass() + "x" + Math.pow(10, 24) + "Kg" + "\n"
            + "Jupiter Mass : " + Jupiter.getMass() + "x" + Math.pow(10, 24) + "Kg" + "\n"
            + Mercury.getPlanetName() + " mass smaller than " + Jupiter.getPlanetName() + " Mass");
} else {
    System.out.println("\nMercury Mass : " + Mercury.getMass() + "x" + Math.pow(10, 24) + "Kg" + "\n"
            + "Jupiter Mass : " + Jupiter.getMass() + "x" + Math.pow(10, 24) + "Kg" + "\n"
            + Jupiter.getPlanetName() + " mass smaller than " + Mercury.getPlanetName() + " Mass");
```

Code to implement polymorphism and displays all planet data using its toString() method. Here super class and super-superclass polymorphism is implemented

Snapshot 4-3:

```
WayPlanet.java | MainDriver.java
History
    }

    //Comparing Mass of Mercury and Jupiter
    if (Mercury.isMassSmaller(Jupiter)) {
        System.out.println("\nMercury Mass : " + Mercury.getMass() + "x" + Math.pow (10, 24) + "Kg" + "\n"
                + "Jupiter Mass : " + Jupiter.getMass() + "x" + Math.pow (10, 24) + "Kg" + "\n"
                + Mercury.getPlanetName() + " mass smaller than " + Jupiter.getPlanetName() + " Mass");
    } else {
        System.out.println("\nMercury Mass : " + Mercury.getMass() + "x" + Math.pow (10, 24) + "Kg" + "\n"
                + "Jupiter Mass : " + Jupiter.getMass() + "x" + Math.pow (10, 24) + "Kg" + "\n"
                + Jupiter.getPlanetName() + " mass smaller than " + Mercury.getPlanetName() + " Mass");
    }

    //Comparing Diameter of Mercury and Jupiter
    if (Mercury.isDiameterGreater(Jupiter)) {
        System.out.println("\nMercury Diameter : " + Mercury.getDiameter() + "Km" + "\n"
                + "Jupiter Diameter : " + Jupiter.getDiameter() + "Km" + "\n"
                + Mercury.getPlanetName() + " diameter greater than " + Jupiter.getPlanetName());
    } else {
        System.out.println("\nMercury Diameter : " + Mercury.getDiameter() + "Km" + "\n"
                + "Jupiter Diameter : " + Jupiter.getDiameter() + "Km" + "\n"
                + Jupiter.getPlanetName() + " diameter greater than " + Mercury.getPlanetName() + " Diameter");
    }

  }
}
```

Displays the result of comparison of mass and diameter between mercury and jupiter planets

5. **Output:**

Note: Initially I had coded to display the output for each instance of MilkyWayPlanet created but I have commented the code currently because it was not specified as the part of the comment. If needed, I can uncomment the code to display the result as shown in Snapshot 5-1

Snapshot 5-1:

```
MainDriver.java | MilkyWayPlanet.java | Planet.java
Source History
30      MilkyWayPlanet Pluto = new MilkyWayPlanet("Pluto", 0.01, 2290, 1, 153.29, 50);
31

output - Assignment1 (run)
run:
Planets Data :
MilkywayplanetsPlanet{Name = Mercury, mass = 33.02x1.0E24Kg, diameter = 4878km, escapeVelocity = 4.25Km/s, revolutionPeriod = 1407.6Hours, meanSurfaceTemperature = 623K}
MilkywayplanetsPlanet{Name = Venus, mass = 4.87x1.0E24Kg, diameter = 12104km, escapeVelocity = 10.4Km/s, revolutionPeriod = 5832.24Hours, meanSurfaceTemperature = 750K}
MilkywayplanetsPlanet{Name = Earth, mass = 5.976x1.0E24Kg, diameter = 12756km, escapeVelocity = 11.2Km/s, revolutionPeriod = 23.93Hours, meanSurfaceTemperature = 293K}
MilkywayplanetsPlanet{Name = Mars, mass = 64.2x1.0E24Kg, diameter = 6794km, escapeVelocity = 5.0Km/s, revolutionPeriod = 24.7Hours, meanSurfaceTemperature = 293K}
MilkywayplanetsPlanet{Name = Jupiter, mass = 0.68x1.0E24Kg, diameter = 142800km, escapeVelocity = 60.3Km/s, revolutionPeriod = 10.0Hours, meanSurfaceTemperature = 163K}
MilkywayplanetsPlanet{Name = Saturn, mass = 0.0569x1.0E24Kg, diameter = 120000km, escapeVelocity = 35.6Km/s, revolutionPeriod = 10.0Hours, meanSurfaceTemperature = 93K}
MilkywayplanetsPlanet{Name = Uranus, mass = 0.87x1.0E24Kg, diameter = 51120km, escapeVelocity = 21.2Km/s, revolutionPeriod = 17.2Hours, meanSurfaceTemperature = 57K}
MilkywayplanetsPlanet{Name = Neptune, mass = 0.0103x1.0E24Kg, diameter = 49528km, escapeVelocity = 23.6Km/s, revolutionPeriod = 16.11Hours, meanSurfaceTemperature = 57K}
MilkywayplanetsPlanet{Name = Pluto, mass = 0.01x1.0E24Kg, diameter = 2290km, escapeVelocity = 1.0Km/s, revolutionPeriod = 153.29Hours, meanSurfaceTemperature = 50K}

Polymorphism(Upcasted to Planet abstract superclass) :
MilkywayplanetsPlanet{Name = Mercury, mass = 33.02x1.0E24Kg, diameter = 4878km, escapeVelocity = 4.25Km/s, revolutionPeriod = 1407.6Hours, meanSurfaceTemperature = 623K}
MilkywayplanetsPlanet{Name = Venus, mass = 4.87x1.0E24Kg, diameter = 12104km, escapeVelocity = 10.4Km/s, revolutionPeriod = 5832.24Hours, meanSurfaceTemperature = 750K}
MilkywayplanetsPlanet{Name = Earth, mass = 5.976x1.0E24Kg, diameter = 12756km, escapeVelocity = 11.2Km/s, revolutionPeriod = 23.93Hours, meanSurfaceTemperature = 293K}
```

output of generated instance of each MilkyWay Planetclass

Snapshot 5-2:

```
MainDriver.java  MilkyWayPlanet.java  Planet.java
Source  History
30          MilkyWayPlanet Pluto = new MilkyWayPlanet("Pluto", 0.01, 2290, 1, 153.29, 50);
31

Output - Assignment1 (run)
Milkywayplanets{Planet{Name = Uranus, mass = 0.87x1.0E24Kg, diameter = 51120km, escapeVelocity = 21.2Km/s, revolutionPeriod = 17.2Hours, meanSurfaceTemperature = 57K}}
Milkywayplanets{Planet{Name = Neptune, mass = 0.0103x1.0E24Kg, diameter = 49528km, escapeVelocity = 23.6Km/s, revolutionPeriod = 16.11Hours, meanSurfaceTemperature = 57K}}
Milkywayplanets{Planet{Name = Pluto, mass = 0.01x1.0E24Kg, diameter = 2290km, escapeVelocity = 1.0Km/s, revolutionPeriod = 153.29Hours, meanSurfaceTemperature = 50K}}

Polymorphism(Upcasted to Planet abstract superclass) :
Milkywayplanets{Planet{Name = Mercury, mass = 33.02x1.0E24Kg, diameter = 4878km, escapeVelocity = 4.25Km/s, revolutionPeriod = 1407.6Hours, meanSurfaceTemperature = 623K}}
Milkywayplanets{Planet{Name = Venus, mass = 4.87x1.0E24Kg, diameter = 12104km, escapeVelocity = 10.4Km/s, revolutionPeriod = 5832.24Hours, meanSurfaceTemperature = 750K}}
Milkywayplanets{Planet{Name = Earth, mass = 5.976x1.0E24Kg, diameter = 12756km, escapeVelocity = 11.2Km/s, revolutionPeriod = 23.93Hours, meanSurfaceTemperature = 293K}}
Milkywayplanets{Planet{Name = Mars, mass = 64.2x1.0E24Kg, diameter = 6794km, escapeVelocity = 5.0Km/s, revolutionPeriod = 24.0Hours, meanSurfaceTemperature = 293K}}
Milkywayplanets{Planet{Name = Jupiter, mass = 0.63x1.0E24Kg, diameter = 142800km, escapeVelocity = 60.0Km/s, revolutionPeriod = 10.0Hours, meanSurfaceTemperature = 163K}}
Milkywayplanets{Planet{Name = Saturn, mass = 0.0569x1.0E24Kg, diameter = 120000km, escapeVelocity = 35.6Km/s, revolutionPeriod = 10.0Hours, meanSurfaceTemperature = 93K}}
Milkywayplanets{Planet{Name = Uranus, mass = 0.87x1.0E24Kg, diameter = 51120km, escapeVelocity = 21.2Km/s, revolutionPeriod = 17.2Hours, meanSurfaceTemperature = 57K}}
Milkywayplanets{Planet{Name = Neptune, mass = 0.0103x1.0E24Kg, diameter = 49528km, escapeVelocity = 23.6Km/s, revolutionPeriod = 16.11Hours, meanSurfaceTemperature = 57K}}
Milkywayplanets{Planet{Name = Pluto, mass = 0.01x1.0E24Kg, diameter = 2290km, escapeVelocity = 1.0Km/s, revolutionPeriod = 153.29Hours, meanSurfaceTemperature = 50K}}

Polymorphism(Upcasted to Object super-superclass) :
Milkywayplanets{Planet{Name = Mercury, mass = 33.02x1.0E24Kg, diameter = 4878km, escapeVelocity = 4.25Km/s, revolutionPeriod = 1407.6Hours, meanSurfaceTemperature = 623K}}
Milkywayplanets{Planet{Name = Venus, mass = 4.87x1.0E24Kg, diameter = 12104km, escapeVelocity = 10.4Km/s, revolutionPeriod = 5832.24Hours, meanSurfaceTemperature = 750K}}
Milkywayplanets{Planet{Name = Earth, mass = 5.976x1.0E24Kg, diameter = 12756km, escapeVelocity = 11.2Km/s, revolutionPeriod = 23.93Hours, meanSurfaceTemperature = 293K}}
Milkywayplanets{Planet{Name = Mars, mass = 64.2x1.0E24Kg, diameter = 6794km, escapeVelocity = 5.0Km/s, revolutionPeriod = 24.0Hours, meanSurfaceTemperature = 293K}}
Milkywayplanets{Planet{Name = Jupiter, mass = 0.63x1.0E24Kg, diameter = 142800km, escapeVelocity = 60.0Km/s, revolutionPeriod = 10.0Hours, meanSurfaceTemperature = 163K}}
Milkywayplanets{Planet{Name = Saturn, mass = 0.0569x1.0E24Kg, diameter = 120000km, escapeVelocity = 35.6Km/s, revolutionPeriod = 10.0Hours, meanSurfaceTemperature = 93K}}
Milkywayplanets{Planet{Name = Uranus, mass = 0.87x1.0E24Kg, diameter = 51120km, escapeVelocity = 21.2Km/s, revolutionPeriod = 17.2Hours, meanSurfaceTemperature = 57K}}
Milkywayplanets{Planet{Name = Neptune, mass = 0.0103x1.0E24Kg, diameter = 49528km, escapeVelocity = 23.6Km/s, revolutionPeriod = 16.11Hours, meanSurfaceTemperature = 57K}}
Milkywayplanets{Planet{Name = Pluto, mass = 0.01x1.0E24Kg, diameter = 2290km, escapeVelocity = 1.0Km/s, revolutionPeriod = 153.29Hours, meanSurfaceTemperature = 50K}}

Mercury Mass : 33.02x1.0E24Kg
Jupiter Mass : 0.63x1.0E24Kg
```

All planets data displayed using toString() method thus implementing polymorphism where objects are upcasted to super and super-superclass.

Snapshot 5-3:

```
19  Milkywayplanets{Planet{Name = Uranus, mass = 86.81x1.0E24Kg, diameter = 51120km, escap
20  Milkywayplanets{Planet{Name = Neptune, mass = 102.4x1.0E24Kg, diameter = 49528km, esca
21  Milkywayplanets{Planet{Name = Pluto, mass = 1.3x1.0E24Kg, diameter = 2290km, escapeVel
22
23  Mercury Mass : 0.3302x1.0E24Kg
24  Jupiter Mass : 1898.0x1.0E24Kg
25  Mercury's mass smaller than Jupiter's mass
26
27  Mercury Diameter : 4879.0km
28  Jupiter Diameter : 139822.0km
29  Jupiter's diameter greater than Mercury's diameter
30
```

Result for comparison of Mercury and Jupiter planets' mass and diameter which has been implemented from interface methods