* **openFile(File inputFile) Method**

initialize an empty returnArrayList

initialize a Scanner object *in* to read inputFile

WHILE (in.hasNextLine())

Add current line to returnArrayList

END WHILE

RETURN returnArrayList

* **createObjects(ArrayList<String> lines) Method**

initialize an empty returnArrayList

FOR EACH String *s* in lines

Initialize a temp String array = s.split(“\t”) that has a length of 4

Add a new Moon(temp[0], Double.parseDouble(temp[1]), Double.parseDouble(temp[2]), Double.parseDouble(temp[3]) to returnArrayList

END FOR

RETURN returnArrayList

* **findMean(ArrayList<Moon> moons, MoonAttributes attribute) Method**

initialize sum = 0

SWITCH (attribute)

CASE RADIUS -> {

FOR EACH Moon M in moons

sum += M.getRadius()

END FOR

RETURN sum / moons.size()

}

CASE DENSITY -> {

FOR EACH Moon M in moons

sum += M.getDensity()

END FOR

RETURN sum / moons.size()

}

CASE DISTANCE -> {

FOR EACH Moon M in moons

sum += M.getDistance()

END FOR

RETURN sum / moons.size()

}

END SWITCH

If cases fail, RETURN -1

* **findHighValue(ArrayList<Moon> moons, MoonAttributes attribute) Method**

initialize max = 0

SWITCH (attribute)

CASE RADIUS -> {

FOR EACH Moon M in moons

IF (M.getRadius() > max)

max = M.getRadius()

END IF

END FOR

RETURN max

}

CASE DENSITY -> {

FOR EACH Moon M in moons

IF (M.getDensity() > max)

max = M.getDensity()

END IF

END FOR

RETURN max

}

CASE DISTANCE -> {

FOR EACH Moon M in moons

IF (M.getDistance() > max)

max = M.getDistance()

END IF

END FOR

RETURN max

}

END SWITCH

If cases fail, RETURN -1

* **findMeanMoon(ArrayList<Moon> moons, MoonAttributes attribute, double meanValue) Method**

SWITCH (attribute)

CASE RADIUS -> {

Search through each Moon in moons and find the radius that is closest to the mean radius

}

CASE DENSITY -> {

Search through each Moon in moons and find the density that is closest to the mean density

}

CASE DISTANCE -> {

Search through each Moon in moons and find the distance that is closest to the mean distance

}

END SWITCH

If cases fail, RETURN new Moon()

* **findLowestMoons(ArrayList<Moon> moons, double value, MoonAttributes attribute) Method**

SWITCH (attribute)

CASE RADIUS -> {

loop to find each Moon whose radius is less than the given value and adds it to the returnArrayList

}

CASE DENSITY -> {

loop to find each Moon whose density is less than the given value and adds it to the returnArrayList

}

CASE DISTANCE -> {

loop to find each Moon whose distance is less than the given value and adds it to the returnArrayList

}

END SWITCH

If cases fail, RETURN new ArrayList

* **outputToFile(String outputMessage, ArrayList<Moon> moons, PrintWriter out) Method**