Note that in your test run adding Venus caused a big jump in area. But then Earth, larger than Venus, caused barely any increase. That should have been a clue that something went wrong.

One problem is that you have 2 "paired" arrays -- planet volumes in 1, planet names in the other. You expect index 0 in each to refer to the same planet. But 1 array goes 0...7 in order of decreasing planet volume, and the other goes in order of increasing distance from the sun. That's why later planets make no ~~the~~ radius diff -- you add biggest volume 1st.

```
/**
 * Ringworld is a class that constructs data that simulates
 * a Ringworld, which is a ring that orbits around a star
 * that is comprised of all of the planets in the universe.
 * Ringworld uses two auxiliary classes, Planet and Converter,
 * to make conversion tasks and planet data easy to access.
 *
 * It's a very bad idea to store the dimensions of the Ringworld
 * in constants because they cannot be converted into other
 * units. So, we will have to make copies of each of the
 * constant instance fields in methods that use them in order
 * to manipulate our units to all agree and be constant.
 *
 * Group Members:
 * Luke Pastore
 * Ansh Motiani
 * Gar Rudnyai
 *
 * @author Gar Rudnyai
 * @version April 17, 2020
 */
public class Ringworld
{
    private double [] planets;
    private double totalVolumeMi3;
    private double RWLengthMi;
    private final double EARTH_SURFACE_AREA_MI2 = 196.94 * Math.pow(10, 6);
    private final double RW_INNER_SURFACE_WIDTH_MI = 9.0 * Math.pow(10, 5);
    private final double RW_INNER_SURFACE_HEIGHT_M = 1.0 * Math.pow(10, 2);
    private final double RW_OUTER_SURFACE_WIDTH_M = 1.0 * Math.pow(10, 2);
    private final double RW_OUTER_SURFACE_HEIGHT_MI = 1.0 * Math.pow(10, 3);

    /**
     * Constructor for Ringworld objects, sets volume and the
     * length of the ringworld to zero
     * @param planets 1D array of volumes of all eight planets
     */
    public Ringworld(double[] planets)
    {
        this.planets = new double[planets.length];
        for(int i = 0; i < this.planets.length; i++)
        {
            this.planets[i] = planets[i];
        }
        this.totalVolumeMi3 = 0;
        this.RWLengthMi = 0;
    }

    /**
     * Returns the volume of a planet in the array of doubles
     * @return the volume of a given planet in cubic miles
     */
    public double getPlanetVolume(int index)
    {
        return Converter.ft3ToMi3(planets[index]);
    }

    /**
     * Adds a specified volume, in cubic miles, to the Ringworld
     * @param amountMi3 amount of volume to add in cubic miles
     */
    public void addVolume(double amountMi3)
```

Why does a Ringworld have an array of planets?
Ok I guess.
Adding planet volumes for name might be clearer code.

```
    {
        this.totalVolumeMi3 += amountMi3;
        double outerSurfaceWidthCopyM = this.RW_OUTER_SURFACE_WIDTH_M;
        double innerSurfaceHeightCopyM = this.RW_INNER_SURFACE_HEIGHT_M;
        this.RWLengthMi = this.totalVolumeMi3 /
            (2 * (Converter.MetersToMi(outerSurfaceWidthCopyM) *
            RW_OUTER_SURFACE_HEIGHT_MI) + (RW_INNER_SURFACE_WIDTH_MI *
            Converter.MetersToMi(innerSurfaceHeightCopyM)));
    }

    /**
     * Returns the radius of the Ringworld in astronomical units
     * @return radius of the Ringworld in astronomical units
     */
    public double getRadius()
    {
        double radiusMi = RWLengthMi / (2 * Math.PI);
        double radiusAU = Converter.MiToAU(radiusMi);
        return radiusAU;
    }

    /**
     * Returns the surface area of the Ringworld in earth units
     * @return the surface area of the Ringworld in earth units
     */
    public double getArea()
    {
        double surfaceAreaMi2 = RW_INNER_SURFACE_WIDTH_MI *
            RWLengthMi;
        double surfaceAreaEarthUnits = Converter.Mi2ToEarthUnits(
            surfaceAreaMi2);
        return surfaceAreaEarthUnits;
    }
}
```

must be wrong too

15/30

design + logic errors