

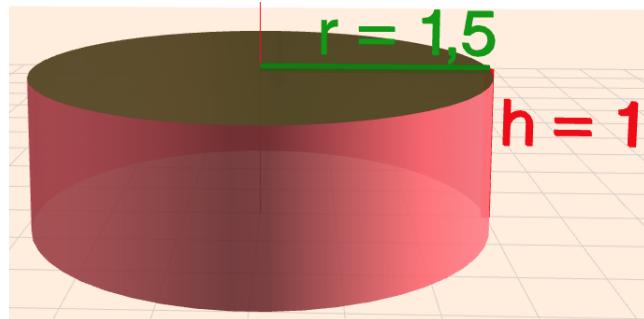
## Puck Class Tests:

### **Table of Contents:**

What is Tested:	Starting Page	Ending Page
Default Constructor	<a href="#">2</a>	<a href="#">2</a>
Constructor 2	<a href="#">3</a>	<a href="#">4</a>
Constructor 3	<a href="#">5</a>	<a href="#">6</a>
getWeight & setWeight	<a href="#">7</a>	<a href="#">7</a>
getDivision & setDivision	<a href="#">8</a>	<a href="#">9</a>
equals	<a href="#">10</a>	<a href="#">10</a>
toString	<a href="#">11</a>	<a href="#">11</a>
compareTo	<a href="#">12</a>	<a href="#">12</a>
menu	<a href="#">13</a>	<a href="#">13</a>
shipPucks	<a href="#">14</a>	<a href="#">19</a>
formatText	<a href="#">20</a>	<a href="#">20</a>

### **Default Constructor Test:**

Puck defaulted:



Weight : ?

Thickness : ?

Radius : ?

Tester:

```
Puck defaulted = new Puck();
```

Should Return:

Weight : 1.00

Thickness : 1.00

Radius : 1.50

Serial : 100000

Result:

Weight : 1.00

Thickness : 1.00

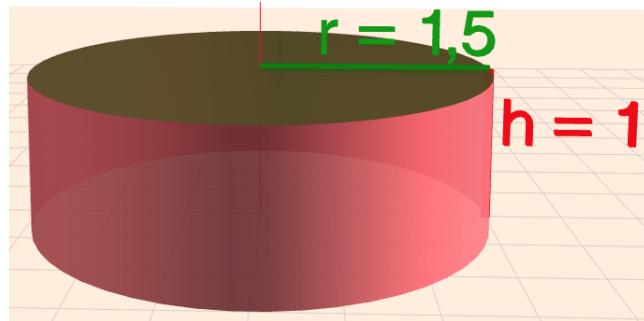
Radius : 1.50

Serial : 100000

---

## Second Constructor Test - Positive Weight:

Puck weighted:



Weight : 7.00

Thickness : ?

Radius : ?

Tester:

```
Puck weighted = new Puck(7.0);
```

Should Return:

Weight : 7.00

Thickness : 1.00

Radius : 1.50

Serial : 100050

Result:

Weight : 7.00

Thickness : 1.00

Radius : 1.50

Serial : 100050

## **Second Constructor Test - Negative Weight:**

Puck antiGraviti:

NO PIC CAUSE CAN'T HAVE NEGATIVE WEIGHT

Weight : -8.00

Thickness : ?

Radius : ?

Tester:

```
Circle c1 = new Puck(-8);  
Puck antiGraviti = (Puck)c1;
```

Should Return:

Weight : 0.00

Thickness : 1.00

Radius : 1.50

Serial : 100100

Result:

Weight : 0.00

Thickness : 1.00

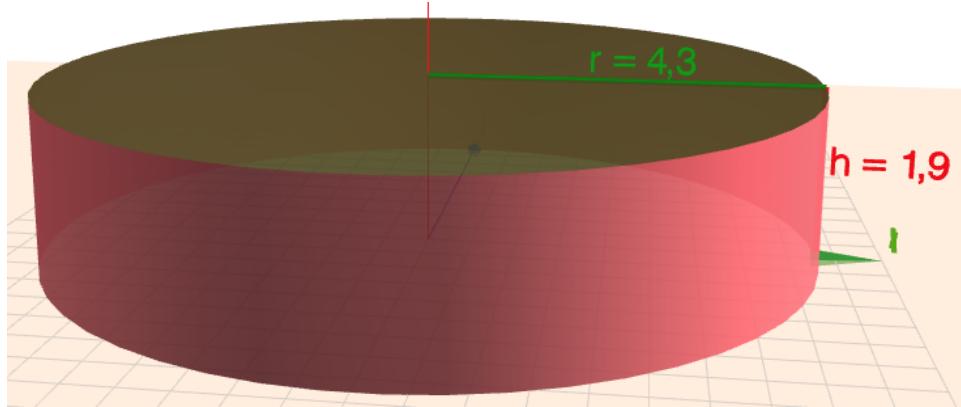
Radius : 1.50

Serial : 100100

---

## **Main Constructor Test - Positives:**

Puck normal:



Weight : 6.2

Radius : 4.3

Thickness : 1.9

Tester:

```
Puck weighted = new Puck(6.2, 4.3, 1.9);
```

Should Return:

Weight : 6.20

Thickness : 4.30

Radius : 1.90

Serial : 100150

Result:

Weight : 6.20

Thickness : 4.30

Radius : 1.90

Serial : 100150

### **Main Constructor Test - Negatives:**

Puck nonExist:

NO PICTURE DUE TO NEGATIVE VALUES.

Weight : -5.4

Radius : -2

Thickness : 0

Tester:

```
Object o1 = new Puck(-5.4, -2, 0);
```

```
Puck nonExist = (Puck)o1;
```

Should Return:

Weight : 0.00

Thickness : 0.00

Radius : 0.00

Serial : 100200

Result:

Weight : 0.00

Thickness : 0.00

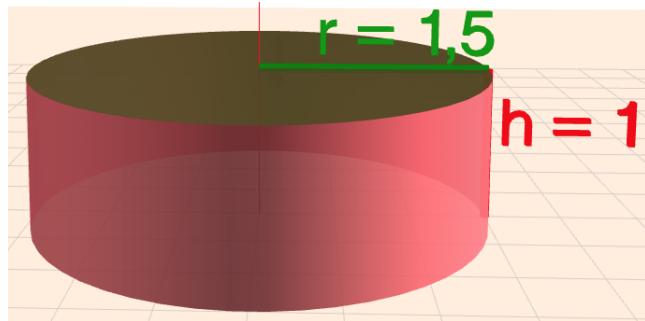
Radius : 0.00

Serial : 100200

---

### 'getWeight' & 'setWeight' Tests:

Puck weightTester:



Weight : 5.9

Radius : 1.5

Thickness : 1

Tester:

```
Disk d1 = new Puck(4.3);
Puck weightTester = (Puck)d1;
weightTester.setWeight(-5.9);
- System.out
  weightTester.setWeight(5.9);
- System.out
```

Should Return:

New Weight: 0.0

New Weight: 5.9

Result:

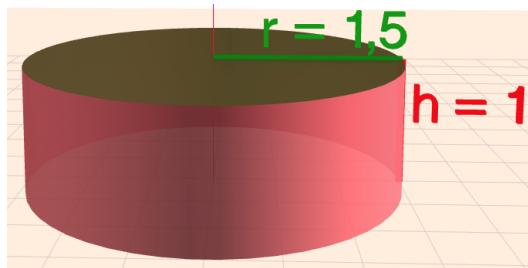
New Weight: 0.0

New Weight: 5.9

---

### 'getDivision' & 'setDivision' Tests - Standard:

Puck divStandart:



Weight : 5.5

Radius : 1.5

Thickness : 1

Weight : 5.7

Radius : 1.5

Thickness : 1

Weight : 6.1

Radius : 1.5

Thickness : 1

Weight : 6

Radius : 1.5

Thickness : 1

Tester:

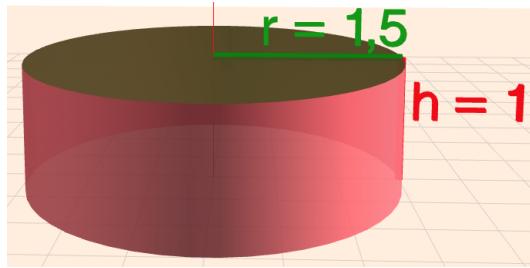
```
Puck divStandart = new Puck(5.5);
- System.out
    divStandart.setWeight(5.7);
- System.out
    divStandart.setWeight(6.1);
- System.out
    divStandart.setWeight(6);
```

Should Return:

```
Puck Division: Standard
Puck Division: Standard
Puck Division: Non-Regulatory
Puck Division: Standard
```

Result:

```
Puck Division: Standard
Puck Division: Standard
Puck Division: Non-Regulatory
Puck Division: Standard
```

**'getDivision' & 'setDivision' Tests - Youth:**Puck divYouth:

Weight : 3.9

Radius : 1.5

Thickness : 1

Weight : 4

Radius : 1.5

Thickness : 1

Weight : 4.2

Radius : 1.5

Thickness : 1

Weight : 4.6

Radius : 1.5

Thickness : 1

Weight : 4.5

Radius : 1.5

Thickness : 1

Tester:

- ```
Puck divYouth = new Puck(3.9);
- System.out
    divStandart.divYouth(4);
- System.out
    divStandart.divYouth(4.2);
- System.out
    divStandart.divYouth(4.6);
- System.out
    divStandart.divYouth(4.5);
- System.out
```

Should Return:

Puck Division: Non-Regulatory  
 Puck Division: Youth  
 Puck Division: Youth  
 Puck Division: Non-Regulatory  
 Puck Division: Youth

Result:

Puck Division: Non-Regulatory  
 Puck Division: Youth  
 Puck Division: Youth  
 Puck Division: Non-Regulatory  
 Puck Division: Youth

## **'equals' Tests:**

### Tester:

- **System.out** (defaulted.equals(defaulted));  
antiGraviti.setWeight(7);
- **System.out** (antiGraviti.equals(weighted));
- **System.out** (divYouth.equals(divStandart));

### Should Return:

Equal? true  
Equal? true  
Equal? false

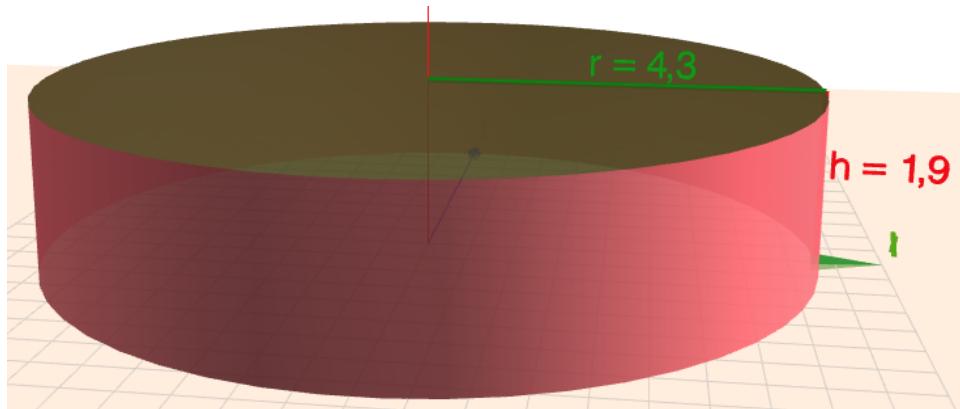
### Result:

Equal? true  
Equal? true  
Equal? false



### **'toString' Test:**

Puck normal:



Weight : 6.2  
Radius : 4.3  
Thickness : 1.9

### Tester:

- **System.out** (normal);

### Should Return:

Radius: 4.30" Thickness: 1.90"  
Weight: 6.20oz  
This puck is a Non-Regulation puck.  
Serial Number 100150

### Result:

Radius: 4.30" Thickness: 1.90"  
Weight: 6.20oz  
This puck is a Non-Regulation puck.  
Serial Number 100150

---

### **'compareTo' Tests:**

Tester:

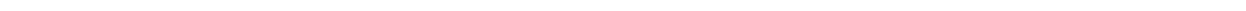
- **System.out** (antiGraviti.equalsTo(weighted));
- **System.out** (nonExist.equals(defaulted));
- **System.out** (defaulted.equals(nonExist));

Should Return:

- 0
- 1
- 1

Result:

- 0
- 1
- 1



**'menu' Test:**

**Tester:**

Radius = 1

Thickness = 4

Weight = 8.4

Amount = 100

**Should Return:**

Radius: 1.00" Thickness: 4.00"

Weight: 8.40oz

This puck is a Non-Regulation puck.

Serial Number 100050

Radius: 1.00" Thickness: 4.00"

Weight: 8.40oz

This puck is a Non-Regulation puck.

Serial Number 105000

**Result:**

Radius: 1.00" Thickness: 4.00"

Weight: 8.40oz

This puck is a Non-Regulation puck.

Serial Number 100050

Radius: 1.00" Thickness: 4.00"

Weight: 8.40oz

This puck is a Non-Regulation puck.

Serial Number 105000

---

**'shipPucks' Tests - 1 Small Package:****Testers:**

1. Radius = 1, Thickness = 1, Weight = 0.5, Amount = 1.
2. R = 1, T = 1, W = 0.5, Amount = 18.
3. R = 1, T = 1, W = 2, Amount = 1.
4. R = 1, T = 1, W = 2, Amount = 18
5. R = 1, T = 1, W = 10, Amount = 1
6. R = 1, T = 1, W = 10, Amount = 3.
7. R = 2, T ≈ 4.588, W = 0.5, Amount = 1.
8. R = 2, T ≈ 4.588, W = 10 = Amount = 1.

**Should ALL Return:**

1xSMALL PKG      \$5.00

**Logic:**

$57.66 \text{ max weight} = 36$   
 $\text{rad} = 1 - S \quad \text{weight} = 0.5 - 10$   
 / thick  
 smallst puck vol =  $3.14 \times 1^2 \times 1 = 3.14$        $\frac{36}{0.5} = 72 \text{ pucks}$   
 based on weight  
 $\frac{57.66}{3.14} = 18 \text{ pucks based on size}$   
 Therefore      rad = 1      weight = 0.5 - 2  
 1 small      = thick = 1  
 amount = 1 - 18       $36 / 18 = 2$   
 ↓  
 max puck vol to fit into small is  
 $1 \times 57.66 \text{ with } 0.5 - \frac{600}{501} \text{ weight.}$   
 $\text{rad} = 2 \quad \text{thick} = 4.588 \quad \frac{57.66}{\text{vol.}}$

(Max weight for last part is 10, because  $10 \times 1 = 10$  which is smaller than 36.)Tot Weight =  $65 / 1 = 65 \approx 10$

**'shipPucks' Tests - 1 Medium Package:**Testers:

1. Radius = 1, Thickness = 1, Weight = 0.5, Amount = 19.
2. R = 1, T = 1, W = 0.5, Amount = 100.
3. R = 1, T = 1, W = 0.65, Amount = 19.
4. R = 1, T = 1, W = 0.65, Amount = 100.
5. R = 1, T = 1, W = 3.42, Amount = 19.
6. R = 1, T = 1, W = 10, Amount = 6.
7. R = 5, T = 4.14, W = 0.5, Amount = 1.
8. R = 5, T = 4.14, W = 10, Amount = 1.

Should ALL Return:

1xMEDIUM PKG \$8.00

Logic:

$mVol = 325.468$  max weight 65 (36+)  
 $rad/thick = 1-5$  weight = 0.5 - 10 amount = 1 - 100

---

smallest puck vol. =  $3.14$        $\frac{65}{0.5} = 130$  pucks based  
 on weight

$\frac{325.468}{3.14} = 103$  pucks based on size (min = 19)

Therefore rad = 1 weight = 0.5 - 0.65  
 1 medium = thick = 1  
 amount = 19 - 100       $65/100 = 0.65$

max puck vol to fit into medium is:  
 1 X 325.468 with 0.5 weight  
 rad = 5 thick = 4.14 = 325.15 vol.

$$\text{Tot Weight} = 65/19 = 3.42$$

(If weight is 1, can't enter more than 65 pucks.)

**'shipPucks' Tests - 1 Large Package:****Testers:**

1. Radius = 5, Thickness = 5, Weight = 0.5, Amount = 1.
2. R = 5, T = 5, W = 0.5, A = 4.
3. R = 5, T = 5, W = 10, A = 1.
4. T = 5, T = 5, W = 10, A = 4.
5. R = 1, T = 1, W = 1, A = 100.
6. R = 1, T = 1, W = 1, A = 66.

**Should ALL Return:**

1xLARGE PKG      \$10.00

**Logic:**

$|V| = 1800 \text{ max weight } 100 \text{ (65+)}$   
 $\text{rad/thick} = 1-5 \text{ weight} = 0.5-10 \text{ amount} = 1-100$

---

smallest puck vol =  $3.14 \frac{100}{0.5} = 200 \text{ pucks based}$   
 $\frac{1800}{3.14} = 573 \text{ pucks based on size (min=100)}$

Therefore no amounts we can enter with min rad, thick & weight will produce 1 large box.

max values:  
 $\text{rad} = 5 \text{ thick} = 5 \text{ vol} = 392.699 \frac{100}{392.699} = 1$   
 $\frac{1800}{392.699} = 4 \text{ pucks} \quad \frac{100}{10} = 10 \text{ pucks (weight)}$

**'shipPucks' Tests - Max Puck Values:****Testers:**

1. Radius = 5, Thickness = 5, Weight = 10, Amount = 5 - 8
2. R = 5, T = 5, W = 10, A = 9 - 12
3. R = 5, T = 5, W = 10, A = 13 - 16
4. R = 5, T = 5, W = 10, A = 41 - 44
5. R = 5, T = 5, W = 10, A = 97 - 100

**Should Return:**

6. 2xLARGE PKG \$20.00
7. 3xLARGE PKG \$30.00
8. 4xLARGE PKG \$40.00
9. 11xLARGE PKG \$110.00
10. 24xLARGE PKG \$240.00
11. 25xLARGE PKG \$250.00

**Logic:**

$$\begin{aligned} \text{rad} &= 5 \quad \text{thick} = 5 \quad \text{weight} = 10 \quad \text{amount} = 5+ \\ \text{2xlarge} &\quad \text{Vol} = 325,058 \\ \text{Vol} &= 392.69 \quad \text{weight}(5) = 50 \quad \frac{1800}{392.69} = 4 \text{ pucks} \end{aligned}$$

$$\begin{aligned} 392.69(4) &= 1570.76 \quad 5-4 = 1 \text{ puck left} \\ 392.69(1) &= 392.69 \rightarrow \text{only 1 puck left} \end{aligned}$$

Since 1 large can only fit 4 pucks at max value every multiple of 4 will signify how many large boxes we have to use.

$\frac{100}{4} = 25$  Therefore, having all vals at max (even amount) will mean we would have to use 25 large packages at most.

Therefore the Total Amount (cost) cannot be ever be above \$476. Max Shipping Cost = \$250

$$\begin{aligned} 250 \\ 2 \text{ per puck} &= 200 \\ \text{box} &= 226 \\ \text{Tot} &= 476. \end{aligned}$$

## **'shipPucks' Tests - Random Values:**

### Testers:

1. Radius = 3.5, Thickness = 2.3, Weight = 1.7, Amount = 10
2. R = 1, T = 5, W = 6, A = 100
3. R = 2.5, T = 1.0, W = 2.5, A = 27
4. R = 2, T = 5, W = 5, A = 57
5. R = 1.5, T = 1, W = 1, A = 30
6. R = 4.6, T = 3.6, W = 2.3, A = 17
7. R = 4.3, T = 2.2, W = 9.0, A = 56
8. R = 4.3, T = 3.4, W = 7.6, A = 36
9. R = 1.3, T = 2.3, W = 4, A = 11
10. R = 3, T = 2, W = 1, A = 36

### Should Return:

1. 1xLARGE PKG \$10.00
2. 6xLARGE PKG & 1xMEDIUM PKG \$68.00
3. 1xLARGE PKG \$10.00
4. 3xLARGE PKG \$30.00
5. 1xMEDIUM PKG \$8.00
6. 3xLARGE PKG \$30.00
7. 5xLARGE PKG & 1xMEDIUM PKG \$58.00
8. 4xLARGE PKG \$40.00
9. 1xMEDIUM PKG \$8.00
10. 1xLARGE PKG & 1xMEDIUM PKG \$18.00

**Note:** There would never be more than 1 medium box / 1 small box since once either of them hits becomes 2 of that type of box, it can easily be converted to the one size above them since it will be more cost efficient to have a small amount of large boxes than many small boxes.

Example:

|                  |         |
|------------------|---------|
| 1 x Large Box    | \$10.00 |
| 2 x Medium Boxes | \$16.00 |
| Total:           | \$26.00 |

Convection:

**2 Medium = 1 Large OR 1 Large & 1 small.**

**1 Large + 1 Large = 2 Large.**

|                 |         |
|-----------------|---------|
| 2 x Large Boxes | \$20.00 |
| Total:          | \$20.00 |

OR

|                 |         |
|-----------------|---------|
| 2 x Large Boxes | \$20.00 |
| 1 x Small Box   | \$ 5.00 |
| Total:          | \$25.00 |

Both are more cost efficient than the original.

---

For 'formatText' compare the result of your entered value to the 'FormattedInvoice.png'  
picture to see if they match.  
The Total Shipping was an added addition.