Assignment: `class TrashCan`

The `TrashCan` class, which you will write, represents a trash can.

`TrashCan` objects are created by calls to a constructor with a `double` parameter that represents the trash can's capacity, which is the maximum weight, in pounds, that a `TrashCan` object can hold. Assume that this value will be greater than or equal to `0`. When a `TrashCan` object is constructed, it is initially empty and its contents have a weight of `0` pounds.

The `TrashCan` class contains an `acceptTrash` method, which has a `double` parameter that represents the weight, in pounds of the trash that will be deposited into the trash can. The amount to be deposited will always be greater than `0` and less that the capacity of the can. If there is already trash in the trash can, the amount to be deposited, combined with the existing trash, may reach or exceed the can's capacity. In this case, the trash can will be filled to its capacity and then emptied before the remaining trash is accepted. The trash can should always be emptied when the weight of the trash in the can reaches the can's capacity. The `acceptTrash` method returns a `double` that represents the number of pounds of trash that can still be added to the trash can before it is completely filled.

The following table contains a sample code execution sequence and the corresponding results. The code execution sequence appears in a class other than `TrashCan`.

Statement	Return Value	Explanation
`double remaining;`		
`TrashCan kitchen = new TrashCan(10.0);`		`TrashCan` object named `kitchen` with a capacity of `10` pounds of trash. Initially there are `0` pounds of trash in the trash can.
`remaining = kitchen.acceptTrash(2.5);`	`7.5`	`kitchen` now holds `2.5` pounds of trash and can take `7.5` more pounds before it needs to be emptied.
`remaining = kitchen.acceptTrash(3.5);`	`4.0`	`kitchen` now holds `6` pounds of trash and can take `4` more pounds before it needs to be emptied.
`remaining = kitchen.acceptTrash(6.0);`	`8.0`	`kitchen` accepts `4` pounds of trash, reaching the `10`-pound capacity, and is then emptied. The remaining `2` pounds of trash are then accepted. `kitchen` now holds `2` pounds of trash and can take `8` more pounds before it needs to be emptied.
`TrashCan bedroom = new TrashCan(3.0);`		`TrashCan` object named `bedroom` with a capacity of `3` pounds of trash. Initially there are `0` pounds of trash in the trash can.
`remaining = bedroom.acceptTrash(1.0);`	`2.0`	`bathroom` now holds `1` pound of trash and can take `2` more pounds before it needs to be emptied.
`remaining = bedroom.acceptTrash(3.2);`	`1.8`	`bathroom` accepts `2` pounds of trash to reach the `3`-pound capacity and then it is emptied. The remaining `1.2` pounds are then accepted. `bedroom` now holds `1.2` pounds of trash and can take `1.8` more pounds before it needs to be emptied.

Assignment: `class TrashCan`

Write the complete `TrashCan` class in the space below. **(total points: 7)**

```
public class TrashCan {

   private double capacity;
   private double trashAmount;

   public TrashCan(double weightCapacity) {
      trashAmount = 0;
      capacity = weightCapacity;
   }

   public double acceptTrash(double dumpWeight) {

      if(trashAmount + dumpWeight >= capacity) {
         trashAmount += dumpWeight - capacity;
      } else {
         trashAmount += dumpWeight;
      }

      return capacity - trashAmount;

}


   public double acceptTrashAlternate(double dumpWeight) {
      trashAmount = (trashAmount + dumpWeight) % capacity;
      return capacity - trashAmount;
   }
```

+1 `class TrashCan;`
no point if: class is `private`, extraneous code outside class; () with or without parameters

+1 `private` instance variables for weight and capacity declared
no point if: `static`, variable outside class, instance variable inside constructor

+1 Constructor `public Trashcan(double ___)`
no point if: `static` or `private`

+1 Constructor initializes capacity to zero (okay if initialized to a value from a parameter)
no point if: assigned from unknown source; assign value to local variable instead of instance variable

+1 Method header `double acceptTrash(double ____)`
no point if: not `public`, wrong method name, incorrect return or parameter type.

+1 Calculates and updates variables used to maintain amount of trash (or remaining space) in the trash can
okay: fail to return correct amount; use an undeclared or variable declared outside the method;
no point if: fail to update the instance variable(s); return something other than remaining capacity

+1 Returns calculated amount remaining from within `acceptTrash`
okay: return an incorrect amount
no point if: fail to return a value in some cases