Programming Problem: Write your own HashMap

See the partially-written BlueJ project, `Hash Map`. This project has three classes:

- `Entry`: Completely written. An object of this class represents a single entry in the map.
- `APHashMap`: Represents a HashMap. You have to write some methods.
- `Tester`: Completely written. You can call `Tester.tester()` to test your `APHashMap`.

Methods in `APHashMap` for you to write

- `private int hash(K key)`

  This returns a hash index for the given key. This value is given by the formula:

  $$H = |h(k)\mathrm{MOD}n|$$

  - $h(k)$ is the built-in Java hash code. You can get this hash code by doing,

    `x.hashCode()`

    on any Java object. This works because `hashCode()` is an instance method in the `Object` class, and every Java object inherits from `Object`.
  - n is the length of the array entries.
  - $|...|$ indicates absolute value (so we don't get a negative index).

  So the idea of this function is to take any Key object, and "convert" it into a valid index in the array.

- `public V put(K key, V value)`

  Works like the familiar `put()` method.

- `public V get(K key)`

  Works like the familiar `get()` method.

- `public V remove(K key)`

  Works like the familiar `remove()` method.

- `public void printMap()`

  Prints each pair in the map.

Sample Terminal output from the method `Tester.tester()`

```
******************************************
* First loop: Adding new items to the map *
******************************************

Adding new key/value pair: (apples, 10)
Map is now:
-----------------
apples, 10
-----------------
Size = 1

Adding new key/value pair: (bananas, 17)
Map is now:
-----------------
apples, 10
bananas, 17
-----------------
Size = 2

Adding new key/value pair: (cherries, 32)
Map is now:
-----------------
apples, 10
cherries, 32
-----------------
Size = 2

Adding new key/value pair: (dates, 24)
Map is now:
-----------------
apples, 10
cherries, 32
dates, 24
-----------------
Size = 3

Adding new key/value pair: (elderberries, 6)
Map is now:
-----------------
elderberries, 6
cherries, 32
dates, 24
-----------------
Size = 3

Adding new key/value pair: (figs, 11)
Map is now:
-----------------
figs, 11
elderberries, 6
cherries, 32
dates, 24
-----------------
Size = 4

Adding new key/value pair: (grapefruit, 3)
Map is now:
-----------------
```

```
figs, 11
grapefruit, 3
cherries, 32
dates, 24
-----------------
Size = 4




After adding all pairs, the state of the map is:
-----------------
figs, 11
grapefruit, 3
cherries, 32
dates, 24
-----------------

*****************************************
* Second loop: Testing the get() method *
*****************************************

apples returns 3
bananas returns 32
cherries returns 32
dates returns 24
elderberries returns 3
figs returns 11
grapefruit returns 3


*****************************************
* Third loop: Testing the remove() method *
*****************************************

Removing apples returns 3
Map is now:
-----------------
figs, 11
cherries, 32
dates, 24
-----------------
Size = 3

Removing bananas returns 32
Map is now:
-----------------
figs, 11
dates, 24
-----------------
Size = 2

Removing cherries returns null
Map is now:
-----------------
figs, 11
dates, 24
-----------------
Size = 2
```

```
Removing dates returns 24
Map is now:
-----------------
figs, 11
-----------------
Size = 1

Removing elderberries returns null
Map is now:
-----------------
figs, 11
-----------------
Size = 1

Removing figs returns 11
Map is now:
-----------------
-----------------
Size = 0

Removing grapefruit returns null
Map is now:
-----------------
-----------------
Size = 0
```