

```
/**
 * A container that can hold at most one T. Thus, a T can only be
 * added if the box is empty.
 *
 * @mathmodel contents : set of Ts
 * @initially contents is empty
 * @constraint |contents| <= 1
 */
public interface Box<T> {

    /**
     * Reports the size of the box. Since the number of elements in the box is
     * at most 1, the method returns either 0 or 1.
     *
     * @ensures size = |contents|
     * @return the number of Ts in the box
     */
    public int size();

    /**
     * Tests whether or not the box contains the particular T.
     *
     * @param target
     *         a T to be found in the box
     * @ensures contains <==> target in contents
     * @return true if and only if the box contains the target
     */
    public boolean contains(T target);

    /**
     * Adds a T to the box. This method is only effective if the box is
     * empty. Otherwise, the box remains unchanged.
     *
     * @param item
     *         a T to be added to the box
     * @alters contents
     * @ensures #contents is empty ==> item in contents <br />
     *         #contents is not empty ==> contents = #contents
     */
    public void insert(T item);

    /**
     * Removes an arbitrary T from the box. Since the box can contain at
     * most one T, there is no ambiguity about which T is removed.
     *
     * @requires |contents| > 0
     * @alters contents
     * @ensures removeAny not in box <br />
     *         removeAny union box = #box
     * @return a T from the box
     */
    public T removeAny();
}
```