Домашно **Collection**

1. Да се създаде имплементация на клас контейнер, който поддържа връзка между обекти тип много към едно. Да се имплементират методите на този клас

```
/**
 * Introduces the notation of many-to-one relation. This is where the M and O of
 * the type signature comes from.
 *
 * Many unique "source" objects refer to one and only "target" object.
 *
 * The class maintains a connection between the target and all the sources that
 * are referring to it.
 *
 * @author Kiril Mitov k.mitov@sap.com
 *
 * @param <M>
 *            the type of the "source" objects.
 * @param <O>
 *            the type of the "target" objects.
 */
public class ManyToOneRelation<M, O> {

	/**
	 * Connects the given source with the given target. If this source was
	 * previously connected with another target the old connection is lost.
	 *
	 * @param source
	 * @param target
	 * @return
	 */
	public boolean connect(M source, O target) {
		return false;
	}

	/**
	 * @param source
	 * @return <code>true</code> if the relation contains the given source
	 */
	public boolean containsSource(M source) {
		return false;
	}

	/**
	 * @param target
	 * @return <code>true</code> if the relation contains the given target
	 */
	public boolean containsTarget(O target) {
		return false;
	}

	/**
	 * @param source
	 * @return the target with which this source is connected
	 */
	public O getTarget(M source) {
		return null;
	}

	/**
```

```
         * @param target
         * @return all the targets that are connected with this source or empty
         *         collection if there are no sources connected with this target.
         */
        public Collection<M> getSources(O target) {
             return null;
        }

        /**
         * Removes the connection between this source and the corresponding target.
         * Other sources will still point to the same target.
         *
         * The target is removed if this was the only source pointing to it and
         * {@link #containsTarget(Object)} will return false.
         *
         * @param source
         */
        public void disconnectSource(M source) {

        }

        /**
         * Removes the given target from the relation. All the sources that are
         * pointing to this target are also removed.
         *
         * If you take the "result" of {@link #getSources(target)} and after that
         * call this method then {@link #containsSource(Object)} will return
         * <code>false</code> for every object in "result".
         *
         * @param target
         */
        public void disconnect(O target) {

        }

        /**
         * @return a collection of the targets.
         */
        public Collection<O> getTargets() {
             return null;
        }

}
```

2. Да се имплементират методите `equals` и `hashCode()`, така че ако две инстанции на ManyToOneRelation съдържат връзки между едни и същи обекти то equals да връща true

```
        public void testEqualsTrue() throws Exception {
             ManyToOneRelation<String, Integer> relation1 = new
ManyToOneRelation<String, Integer>();
             ManyToOneRelation<String, Integer> relation2 = new
ManyToOneRelation<String, Integer>();
             relation1.connect("Integer1", new Integer(1));
             relation1.connect("Integer2", new Integer(2));

             relation2.connect("Integer1", new Integer(1));
             relation2.connect("Integer2", new Integer(2));

             relation1.equals(relation2) - should be true
```

```
            relation1.hashCode() == relation2.hashCode() - should be true
      }



public void testEqualsFalse() throws Exception {
      ManyToOneRelation<String, Integer> relation1 = new ManyToOneRelation<String,
Integer>();
      ManyToOneRelation<String, Integer> relation2 = new ManyToOneRelation<String,
Integer>();
      relation1.connect("Integer1", new Integer(1));

      relation2.connect("Integer1", new Integer(1));
      relation2.connect("Integer2", new Integer(2));

      relation1.equals(relation2) - should be false
}
```