

```
package ast;

import java.util.ArrayList;

/**
 * Class that extends from Elem, that stores information about a method or constructor
 * @author Sergio Alcocer
 *
 */
public class Method extends Elem {

    private boolean isConstructor;
    private String returnType;
    private ArrayList<String> params;

    /**
     * Constructor of the class
     */
    public Method() {
        super(T_METHOD);
        isConstructor = false;
        params = new ArrayList<String>();
    }

    /**
     * Adds a parameter to the definition of the method
     * @param _paramType type of the parameter added
     */
    public void addParam(String _paramType) {
        params.add(_paramType);
    }

    /**
     * Sets the current method as a constructor
     */
    public void setAsConstructor() {
        isConstructor = true;
    }

    /**
```

```
* Sets the returned type of the method
* @param _type returned type to be set
*/
public void setReturnedType(String _type){
    returnedType = _type;
}

/**
 * Gets the returned type of the method
 * @return returned type of the method
 */
public String getReturnedType(){
    return new String(returnedType);
}

/**
 * Gets if the method has been set as constructor
 * @return true if the method is a constructor, false in other case
 */
public boolean isConstructor(){
    return isConstructor;
}

/**
 * Displays the description of the method
 * @return description of the method to be shown in the list
 */
public String toString(){
    String returned;
    if(access == PRIVATE){
        returned = new String("- ");
    }else{
        returned = new String("+ ");
    }
    returned += returnedType + " ";

    if (!isConstructor){
        returned += name + " ";
    }
}
```

```
        returned += "( ";

        if(params.size() > 0){
            returned += params.get(0);
            for(int i = 1; i < params.size(); i++){
                returned += ", " + params.get(i);
            }
        }
        returned += " )";

        return returned;
    }

    /**
     * Gets the number of parameters
     * @return number of parameters
     */
    public int getParamNumber(){
        return params.size();
    }

    /**
     * Gets a single parameter
     * @param paramNumber position of the parameter to be retrieved
     * @return type of the parameter retrieved
     */
    public String getParam(int paramNumber){
        return params.get(paramNumber);
    }
}
```