**Programming by Contract**

**Programming by Contract** is a concept which creates a contract between the software developer and the software user.

Every method starts with a **Pre-condition** that must be satisfied by the user of the feature.

* They don’t have to be handled by the developer
* If they hold, error checking is redundant
* They can be specified in Javadoc along with associated parameters @param @pre
* **/\*  
   \* Method to deposit money into a bank account  
   \* @param amount is the amount to be deposited  
   \* @pre depositAmount > 0  
   \*/**

Every method ends with a **Post-condition** which the developer guarantees to be true (IFF pre-conditions were met when the method is called).

* The developer guarantees expected behaviour for the end user
* **/\*   
   \* Method to deposit money into a bank account  
   \* @post** **balance = balance + amount  
   \*/**

Each class has an **Invariant** which is a logical condition that must be satisfied after any changes are made to the object represented by the class. It guarantees that the object is in a valid state.

* The condition is true after the constructor has completed execution (guarantees no invalid objects are created)
* The condition is true before and after a method call, but it can be temporarily violated during execution of a method.
* **/\*   
   \* Bank Account handles the deposit and withdrawal of funds  
   \* @invariant balance >= 0  
   \*/**

Getters don’t really need @pre @post as no state is being changed in the object. The user is simply viewing the object.