IDefn.java

```java
// Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas

package jminusminus;

/**
 * The IDefn type is used to implement definitions of those things (local
 * variables, formal arguments, types) that are named in some context (or
 * scope).
 */

interface IDefn {

    /**
     * The (local variable, formal parameter, or local or imported name)
     * definition's type.
     * 
     * @return the definition's type.
     */

    public Type type();

}

/**
 * A definition of a type name. In the first instance, an identifier, but later
 * resolved to a local name or an imported name.
 */

class TypeNameDefn implements IDefn {

    /** The definition's type. */
    private Type type;

    /**
     * Construct a type name definition given its type.
     * 
     * @param type
     *            the definition's type.
     */

    public TypeNameDefn(Type type) {
        this.type = type;
    }

    /**
     * The type for this definition.
     * 
     * @return the definition's type.
     */

    public Type type() {
        return type;
    }

}

/**
 * The definition for a local variable (including formal parameters). All local
 * variables are allocated on the stack at fixed offsets from the base of the
 * stack frame, and all have types. Some local variables have initializations.
 */

class LocalVariableDefn implements IDefn {

    /** The local variable's type. */
    private Type type;

    /**
     * The local variable's offset from the base of the current the stack frame.
     */
    private int offset;

    /** Has this local variable been initialized? */
    private boolean isInitialized;

    /**
     * Construct a local variable definition for a local variable.
     * 
     * @param type
     *            the variable's type.
     * @param offset
     *            the variable's offset from the base of the current stack frame
     *            (allocated for each method invocation.)
     */

    public LocalVariableDefn(Type type, int offset) {
        this.type = type;
        this.offset = offset;
    }

    /**
     * The type for this variable.
     * 
     * @return the type.
     */

    public Type type() {
        return type;
    }

    /**
     * The offset of this variable on the stack frame.
     * 
     * @return the offset.
     */

    public int offset() {
        return offset;
    }

    /**
     * Initialize this local variable.
     */

    public void initialize() {
        this.isInitialized = true;
    }

    /**
     * Has this local variable been initialized?
     * 
     * @return true or false.
     */

    public boolean isInitialized() {
        return isInitialized;
    }

}

```