Class Types has static methods for type equivalencing and other forms of type interrogation. Note that this class must be compiled with a language-specific sym.java file. It therefore must be copied in source form and compiled together with a specific parser, since it cannot be compiled in with a the stand-alone a4-support.jar files.

```java
public class Types {
    /**
     * Return true if t1 and t2 are structurally equivalent.
     */
    public static boolean equiv(TypeNode t1, TypeNode t2) {
        return samePrimitiveTypes(t1, t2) // ||/* perform parallel recursive descent on type structures */;
    }

    /**
     * Return true if t1 and t2 are name equivalent.
     */
    public static boolean equivName(TypeNode t1, TypeNode t2) {
        return samePrimitiveTypes(t1, t2) || sameIdentTypes(t1, t2);
    }

    /**
     * Return true if t1 and t2 are the same primitive types.
     */
    public static boolean samePrimitiveTypes(TypeNode t1, TypeNode t2) {
        return isInt(t1) && isInt(t2) || isFloat(t1) && isFloat(t2) || isString(t1) && isString(t2) || isBool(t1) && isBool(t2);
    }

    /**
     * Return true if the given type is an atomic integer type. This is the case if the TypeNode id = INT or if the id is IDENT and its string ident value is "integer". This supports languages in which the integer type is designated by a keyword, as well as languages where it is designated by a pre-defined identifier named "integer".
     */
    public static boolean isInt(TypeNode t) {
        return (t.id == sym.INT) || (t.id == sym.IDENT) && (((LeafNode) t.child1).value).equals("integer");
    }

    /**
     * Return true if the given type is an atomic floating point type. This is the case if the TypeNode id = FLOAT or if the id is IDENT and its string ident value is "real". This supports languages in which the integer type is designated by a keyword, as well as languages where it is designated by a pre-defined identifier named "real".
     */
    public static boolean isFloat(TypeNode t) {
        return (t.id == sym.FLOAT) || (t.id == sym.IDENT) && (((LeafNode) t.child1).value).equals("real");
    }

    /**
     * Return true if the given type is an atomic string type. This is the case if the TypeNode id = STRING or if the id is IDENT and its integer ident value is "string". This supports languages in which the integer type is designated by a keyword, as well as languages where it is designated by a pre-defined identifier named "string".
     */
    public static boolean isString(TypeNode t) {
        return (t.id == sym.STRING) || (t.id == sym.IDENT) && (((LeafNode) t.child1).value).equals("string");
    }
}
```
/**
 * Return true if the given type is an atomic floating point type. This is
 * the case if the TypeNode id = BOOLEAN or if the id is IDENT and its
 * string ident value is "boolean". This supports languages in which the
 * integer type is designated by a keyword, as well as languages where it
 * is designated by a pre-defined identifier named "boolean".
 */

public static boolean isBool(TypeNode t) {
    return
    (t.id == sym.BOOLEAN) ||
    (t.id == sym.IDENT) &&
    (((LeafNode) t.child1).value).equals("boolean");
}

/**
 * Return true if the given type is an atomic floating point type. This is
 * the case if the TypeNode id = VOID or if the id is IDENT and its string
 * ident value is "void". This supports languages in which the integer
 * type is designated by a keyword, as well as languages where it is
 * designated by a pre-defined identifier named "void".
 */

public static boolean isVoid(TypeNode t) {
    return
    (t.id == sym.VOID) ||
    (t.id == sym.IDENT) &&
    (((LeafNode) t.child1).value).equals("void");
}