Reflection Examples

URL

```
URL url = new URL("https", "ualberta.ca", "/faculties");

String urlString = url.toExternalForm();
```

Reflection Counterpart

- Database stores different object in different tables
- Loader does not need to have intimate knowledge about types
- Config File:
  - Class = “java.net.URL”
  - Stringfy = “toExternalForm”

Reflection Counterpart

```
Class<?> type = Class.forName("java.net.URL");
```

Reflection Counterpart

```
Constructor<?> constructor = type.getConstructor(
    new Class[]{String.class, String.class, String.class});
```

Reflection Counterpart

```
Object instance =
  constructor.newInstance("https", "ualberta.ca", "/faculties");
```

Reflection Counterpart

```
Class<?> type = Class.forName("java.net.URL");
```

```
Method method = type.getMethod("toExternalForm");
```

Reflection Counterpart

```
Object methodCallResult = method.invoke(instance);
```

@Test

- How does JUnit know?

```
<T extends Annotation> T getAnnotation(Class<T> annotationType);
```

```
Test t = method.getAnnotation(Test.class);
```

@interface Test {…}

Useful Functions

- Object: public final Class<?> getClass()
  - Returns the runtime class of this Object. The returned Class object is the object that is locked by static synchronized methods of the represented class.
- Class<T>: public Class<?> getComponentType()
  - Returns the Class representing the component type of an array. If this class does not represent an array class this method returns null.
- Class<T>: public boolean isArray()
- Array: public static int getLength(Object array)
- Array: public static Object newInstance(Class<?> componentType, int length)
- Array: public static Object get(Object array,int index)
- System: public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)
- PrintStream: public void print(Object obj)