Random Testing

+

Delta Debugging

Test Case Generator

```
class A {
  public A(int x){
  public int m1(){...}
  public int m2(B b){...}
  private int m3(){...}
}
```

Generated Test

```
class  ATest {
@Test
public generatedTest(){
   A v1 = new A(5236);
   v1.m1();
   B v2 = new B();
   v1.m2(v2);
   A v3 = new A(-7829);
   v3.m2(v2);
   ...
```

Test Case Generator

- Use reflection to get constructors and methods, and their parameters.
- Use constructor objects to create an instance.
- Use method objects to call a method.
- If the call succeeded produce equivalent Java code.

Object Pool

- Returns objects for given class/interface.
- Allows reuse of objects.

Delta Debugging

- Start with a failing test.
- Apply ddmin to the the source code of the test.
- Compile and load intermediate versions of the class. (see tests)

Class Loading in Java

- Class loading: Loading the binary representation into the JVM.
- ClassLoaders load a class.
- Each class is uniquely defined by its ClassLoader and its name.
- Classes are loaded at the first active use or explicitly with a call to loadClass().

Bootstrap Class Loader

- Loads bootstrap classes, e.g. classes from rt.jar/classes.jar (containing java.lang classes) and from the given classpath.
- Order on classpath matters.

Own Class Loaders

- Java uses a delegation model for loading classes.
- Each class loader has a parent. (except bootstrap class loader)
- Only load a class when it is not already loaded by parent. Otherwise findClass is called.

Pitfalls

- Endless Recursion: circular dependency.
- Endless loops: no method is callable.