Static Checking of Safety Critical Java Annotations

Daniel Tang, Ales Plsek, Jan Vitek

S3 Lab, Purdue University

http://www.ovmj.net/oscj/

Open Safety-Critical Java

Safety Critical Java

- High level story: Java for safety critical systems
    - Safety critical systems may cause harm to persons if they fail, so they require vigorous certifications
- Java annotations may help the certification process
    - A mechanism for adding metadata to Java constructs for compile-time or run-time processing
    - Java annotations preserved in the bytecode as well
    - Enhanced in Java 7 by allowing use in more constructs, enabling construction of pluggable type systems in Java

API Visibility

```
public abstract class PeriodicEventHandler
        extends ManagedEventHandler
        implements Runnable {

  public PeriodicEventHandler(...) ...

  public ReleaseParameters getReleaseParameters() ...

  public final void run() {}
```

API Visibility

```
@SCJAllowed(LEVEL_0, members=true)
class M extends CyclicExecutive {

    public void initialize() {
       PEH p = new PEH(...);
       p.run();
       ...

@SCJAllowed(LEVEL_0, members=true)
class PEH extends PeriodicEventHandler {
 public ReleaseParameters getReleaseParameters(){
        ...
 }
```

API Visibility

```
@SCJAllowed(LEVEL_0)
public abstract class PeriodicEventHandler
      extends ManagedEventHandler
      implements Runnable {

  public PeriodicEventHandler(...) ...

@SCJAllowed(LEVEL_2)
public ReleaseParameters getReleaseParameters() ...

@SCJAllowed(INFRASTRUCTURE)
public final void run() {}
```

Level Compliance Annotations

	parameter	values
@SCJAllowed	value	LEVEL_0 LEVEL_1 LEVEL_2 SUPPORT INFRASTRUCTURE HIDDEN (default)
	members	TRUE FALSE (default)

Phase Restrictions

```
class M extends CyclicExecutive {
  public void setUp() {...}
  public void tearDown() {...}
}
```

```
@SCJAllowed(LEVEL_0, members=true)
class PEH extends
PeriodicEventHandler {
  public void handleEvent() {
    new PEH(...);
    ...
    getCurrentMission().tearDown();
```

Phase Restrictions

```
class M extends CyclicExecutive {
  ...
  @SCJRestricted(INITIALIZATION)
  public void setUp() {...}
  @SCJRestricted(CLEANUP)
  public void tearDown() {...}
}
@SCJAllowed(LEVEL_0, members=true)
class PEH extends PeriodicEventHandler {
  SCJRestricted(EXECUTION)
  public void handleEvent() {
    new PEH(...);
    ...
    getCurrentMission().tearDown();
...
```

Phase Annotations

	parameters	values
@SCJRestricted	value	INITIALIZATION RUN CLEANUP ALL (default)

Rules:

- element of level x may be used only in code that has level x or higher
- it is illegal to override a method to change compliance
- a subclass must not have a higher compliance level that its superclass

Behavior Restrictions

```
@SCJAllowed(LEVEL_1)
public class IH extends InterruptHandler{

  @SCJRestricted(mayAllocate=false,
                 maySelfSuspend=false)
  protected void handleInterrupt() {
     foo();
  }

  protected void foo() {
    new PEH(...);
    sleep(); ...
```

Behavior Restrictions

```
@SCJAllowed(LEVEL_1)
public class IH extends InterruptHandler{

   @SCJRestricted(mayAllocate=false,
                  maySelfSuspend=false)
   protected void handleInterrupt() {
      foo();
   }

   @SCJRestricted(mayAllocate=false,
                  maySelfSuspend=false)
   protected void foo() {
       new PEH(...);
       sleep();
```

Behavior Restrictions

	parameters	values
@SCJRestricted	mayAllocate	TRUE (default)
		FALSE
	maySelfSuspend	TRUE
		FALSE (default)

Rules:

- element of level x may be used only in code that has level x or higher
- it is illegal to override a method to change compliance
- a subclass must not have a higher compliance level that its superclass

Memory Safety

```
class PEH extends PeriodicEventHandler {
  Data data;
  public void handleEvent() {
    R r = new R(this);
    ManagedMemory.getCurrentManagedMemory().
                  enterPrivateMemory(3000, r); ...

class R implements Runnable {
  PEH p;
  public void run() { p.data = new Data(); }
}
class Data { ... }
```

Memory Safety

```
class PEH
  Data data;

    R r = new R
```

```
                    enterPrivateMemory(     r)
```

```
class R
  PEH p;
                          p.data = new Data()
```

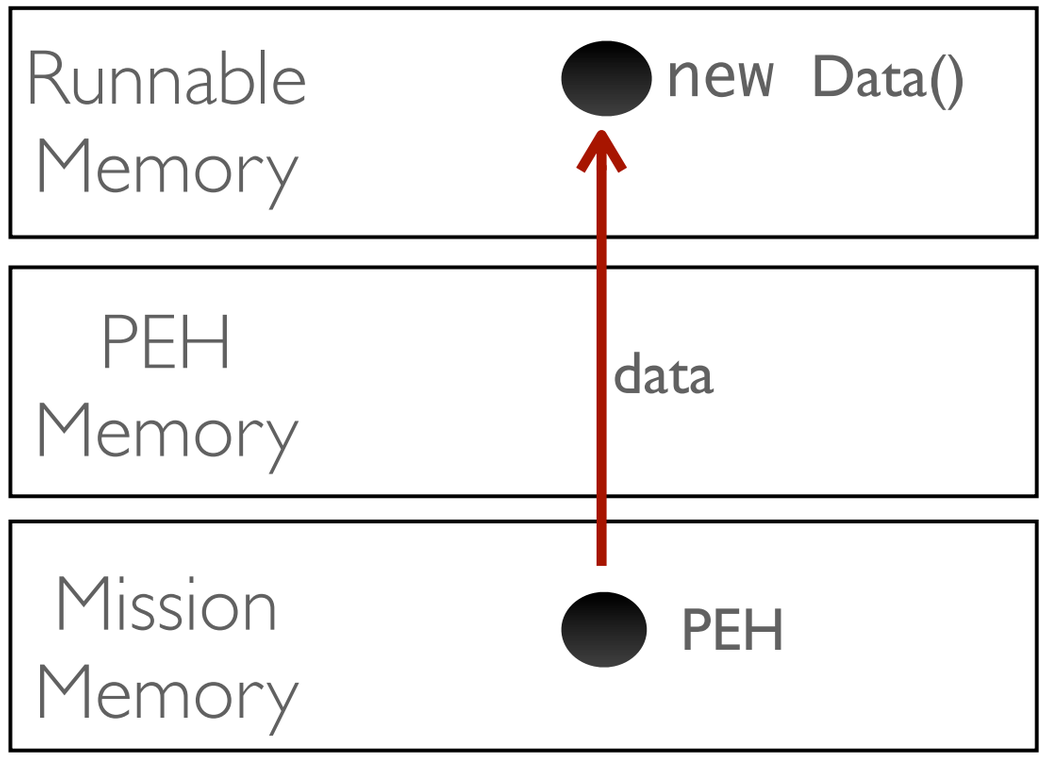
```
class Data { ... }
```

Memory Safety

```
class PEH
 Data data;
  R r = new R
    enterPrivateMemory(r)

class R
  PEH p;
   p.data = new Data()

class Data
```



Memory Safety

```
@Scope("M") @RunsIn("PEH")
class PEH extends PeriodicEventHandler {
  Data data;
  public void handleEvent() {
    @DefineScope(name="R", parent="H")
    R r = new R(this);
    ManagedMemory.getCurrentManagedMemory().
                  enterPrivateMemory(3000, r); ...

@Scope("PEH") @RunsIn("R")
class R implements Runnable {
  PEH p;
  public void run() { p.data = new Data(); }
}
@Scope("R") class Data { ... }
```

Memory Safety

```
@Scope("M") @RunsIn("PEH")
class PEH extends PeriodicEventHandler {


    @DefineScope(name="R", parent="H")
    R r = new R(this);


@Scope("PEH") @RunsIn("R")
class R implements Runnable {


@Scope("R") class Data { ... }
```

Memory Safety Annotations

	parameters	values
@DefineScope	name	a name of newly defined scope
	parent	a parenting scope of a new scope
@Scope	name	a name of scope in which the object is allocated
@RunsIn	name	name of scope where a method will allocate

@Scope

```
@Scope("M") @RunsIn("PEH")
class PEH extends PeriodicEventHandler {

   ... new Data() ....
}
```

```
@Scope("PEH") @RunsIn("R")
class R implements Runnable {
 ... new Data() ...
}

@Scope("R") class Data { ... }
```

Objects with no @Scope

```
@Scope("M") @RunsIn("PEH")
class PEH extends PeriodicEventHandler {

    ... new Data() ....
}
```

```
@Scope("PEH") @RunsIn("R")
class R implements Runnable {
  ... new Data() ...

}

class Data { ... }
```

Class and Fields

```
@Scope("M") @RunsIn("PEH")
class Clazz {
    Field f;
    Data  d;

    @RunsIn("R")
    public void foo(Data d) {
        this.d = d;
    }
}

@Scope("R") class Field { ... }

class Data { ... }
```

Field must be in the same or parent scope

d may not reside in immortal

no @Scope annotation

Class Casting

```
@Scope("PEH") @RunsIn("R")
class R implements Runnable {
  ... Foo f = (Foo) new Data() ...
}
```

```
class Foo { ... }

@Scope("R") class Data extends Foo { ... }
```

Enter Child Scope

```
@Scope("M") @RunsIn("PEH")
class PEH
    ...
    @DefineScope(name="R", parent="PEH")
    R r = new R(this);
    ManagedMemory.getCurrentManagedMemory().
                  enterPrivateMemory(3000, r); ...

    @DefineScope(name="R2", parent="PEH")
    R r2 = new R(this); ....


@Scope("PEH") @RunsIn("R")
class R implements Runnable { ...}
```

Conclusion

- Checker Implementation
    - Java 7 Checker Framework
    - Compile-time checking (Eclipse plugin coming soon)
- Evaluation
    - miniCDj benchmark Case Study
        - ~100 annotations, ~100 examples in the Checker distribution
    - @SCJAllowed and @SCJRestricted easy to use
    - Memory safety annotations
    - Sometimes overly restrictive, resulting in class duplication