ISEL

Languages and Managed Runtimes

2022

Week 6 – meta-programming

Outline

- Remember – Reflection
- Meta-programming
- Remember – Logger
- LoggerDynamic

Remember - Reflection

Ability to **introspect**, the structure and behavior of a program at **runtime**.

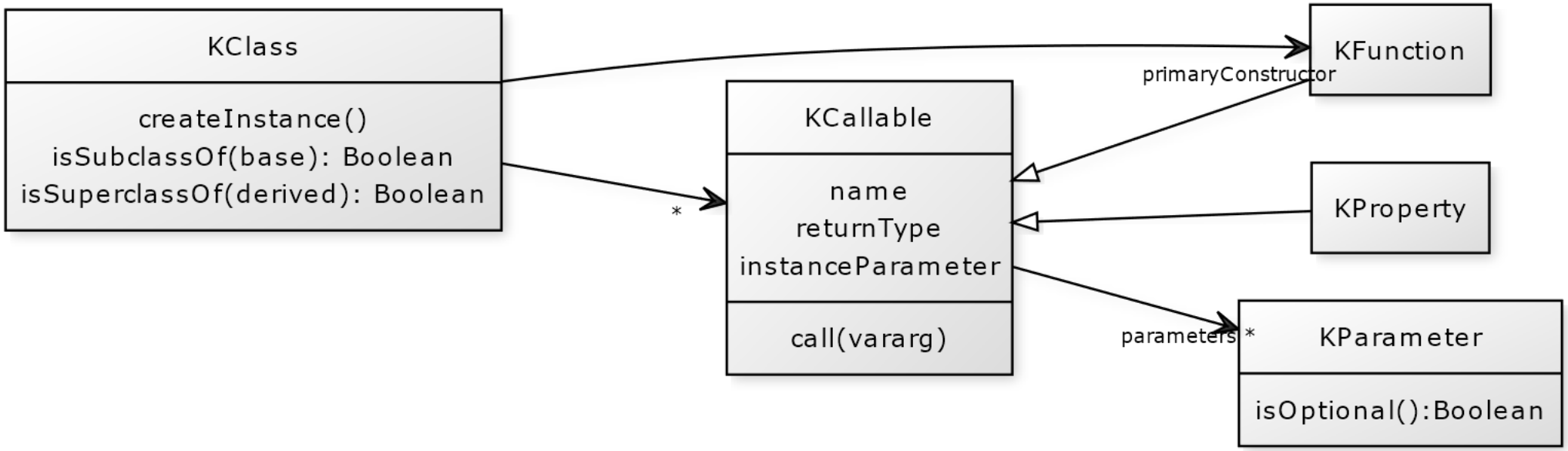
Examples:

- Kotlin - kotlin.reflect
  e.g. `foo::class.declaredFunctions.find{ it.name == "hello"}?.call(foo)`

- Java - java.lang.reflect
  e.g. `foo.getClass().getDeclaredMethod("hello").invoke(foo)`

- JavaScript
  e.g. `foo['hello']()`
- `...`

Kotlin - kotlin.reflect



What can you do?

> E.g. Logger, log4j, Javap, intellisense, Unit tests library (Junit), ORM, dependency injection (Spring), etc

What can you **not** do ?

> Modify or **create new Types dynamically** (at runtime).

Outline

- Remember – Reflection
- Meta-programming
- Remember – Logger
- LoggerDynamic

Meta-programming

Ability to read, generate, transform or modify programs **dynamically** (at runtime).

Examples:

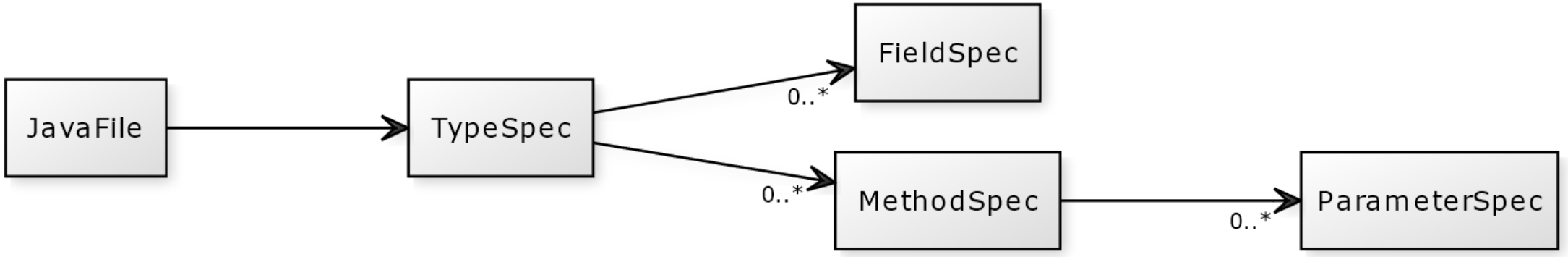
- .net - System.Reflection.Emit

- Java – third parties, e.g. Javassist (Java **bytecode** engineering toolkit), ASM (Java **bytecode** manipulation and analysis framework), and others.

- Javascript

  e.g. `foo.hello = function() { console.log('hello') }`

JavaPoet




```
val myDynamic = TypeSpec
    .classBuilder("MyDynamic")
    .addField(fieldNr)
    .addMethod(ctor)
    .addMethod(mul)
    ....
    .build()

va file = JavaFile.builder(“pt.isel", myDynamic).build()
```

java.lang.Class

```
val fieldNr = FieldSpec.builder(Int::class.java, "nr")....build()

val ctor = MethodSpec.constructorBuilder()....build()

val mul = MethodSpec.methodBuilder("mul").....build()
```

Example

```java
public final class MyDynamic {
  final int nr;

  public MyDynamic(int nr) {
    this.nr = nr;
  }
  public int mul(int other) {
    return this.nr * other;
  }
}
```

```
val myDynamic = TypeSpec
    .classBuilder("MyDynamic")
    .addField(fieldNr)
    .addMethod(ctor)
    .addMethod(mul)
    ....
    .build()

va file = JavaFile.builder(“pt.isel", myDynamic).build()
```

```
val fieldNr = FieldSpec.builder(Int::class.java, "nr")....build()

val ctor = MethodSpec.constructorBuilder()....build()

val mul = MethodSpec.methodBuilder("mul").....build()
```

Compilation and class load

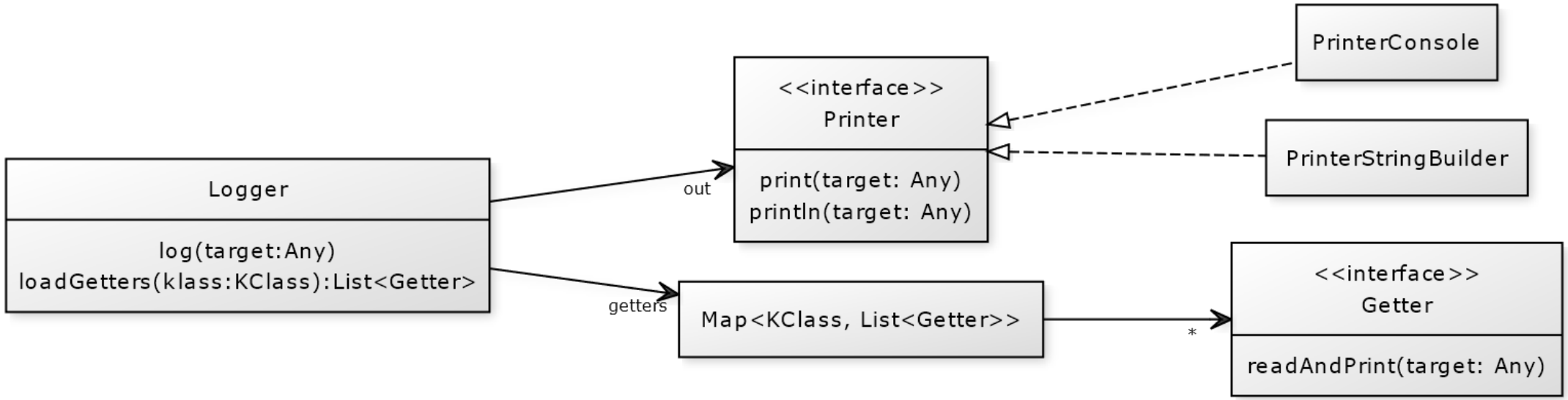
```
fun loadAndCreateInstance(source: JavaFile, vararg args: Any): Any {
   // Save source in .java file.
   source.writeToFile(root)

   // Compile source file.
   ToolProvider
    .getSystemJavaCompiler()
    .run(null, null, null, "${root.path}/${source.typeSpec.name}.java")

   // Load and instantiate compiled class.
   return classLoader
      .loadClass(source.typeSpec.name)
      .declaredConstructors[0]
      .newInstance(*args)
}
```

```
val root = File("./build")
val classLoader = URLClassLoader.newInstance(arrayOf(root.toURI().toURL()))
val compiler = ToolProvider.getSystemJavaCompiler()
```

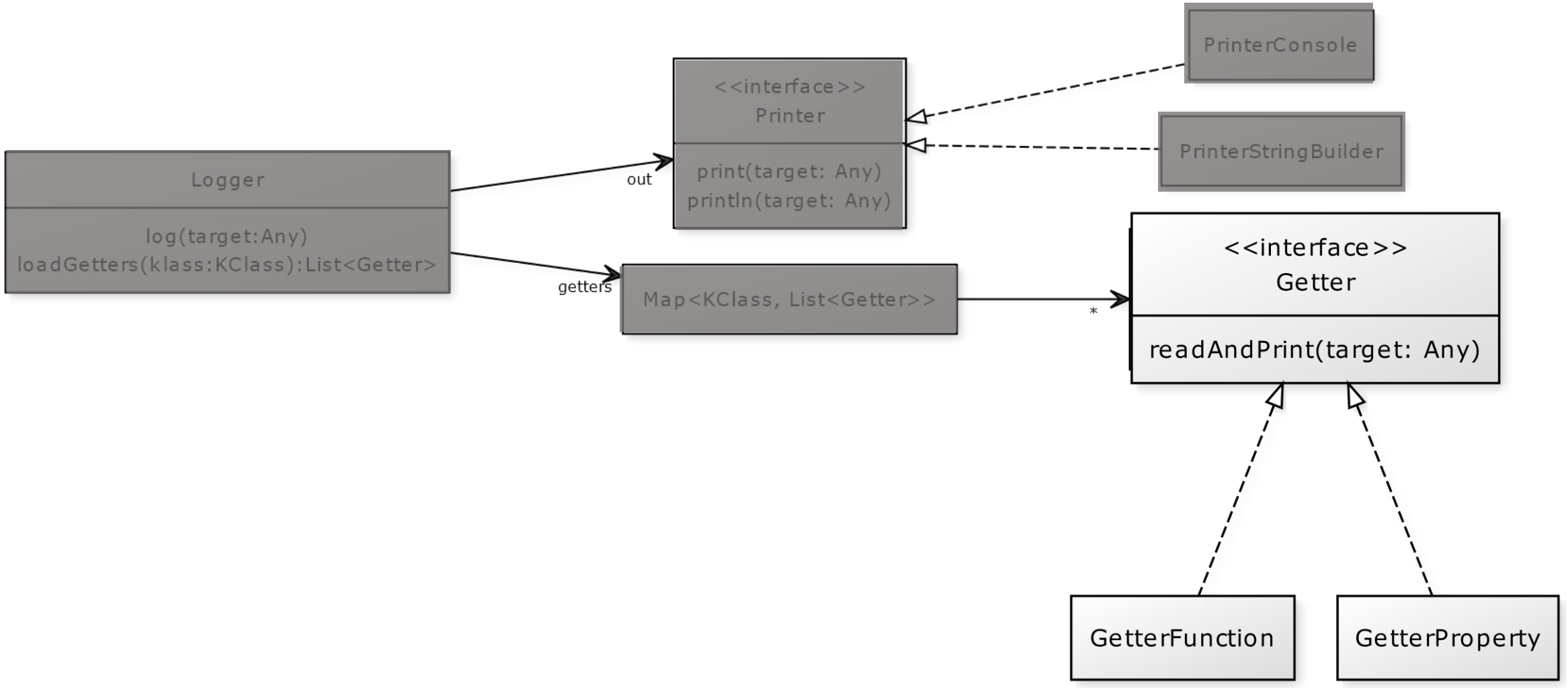
Outline

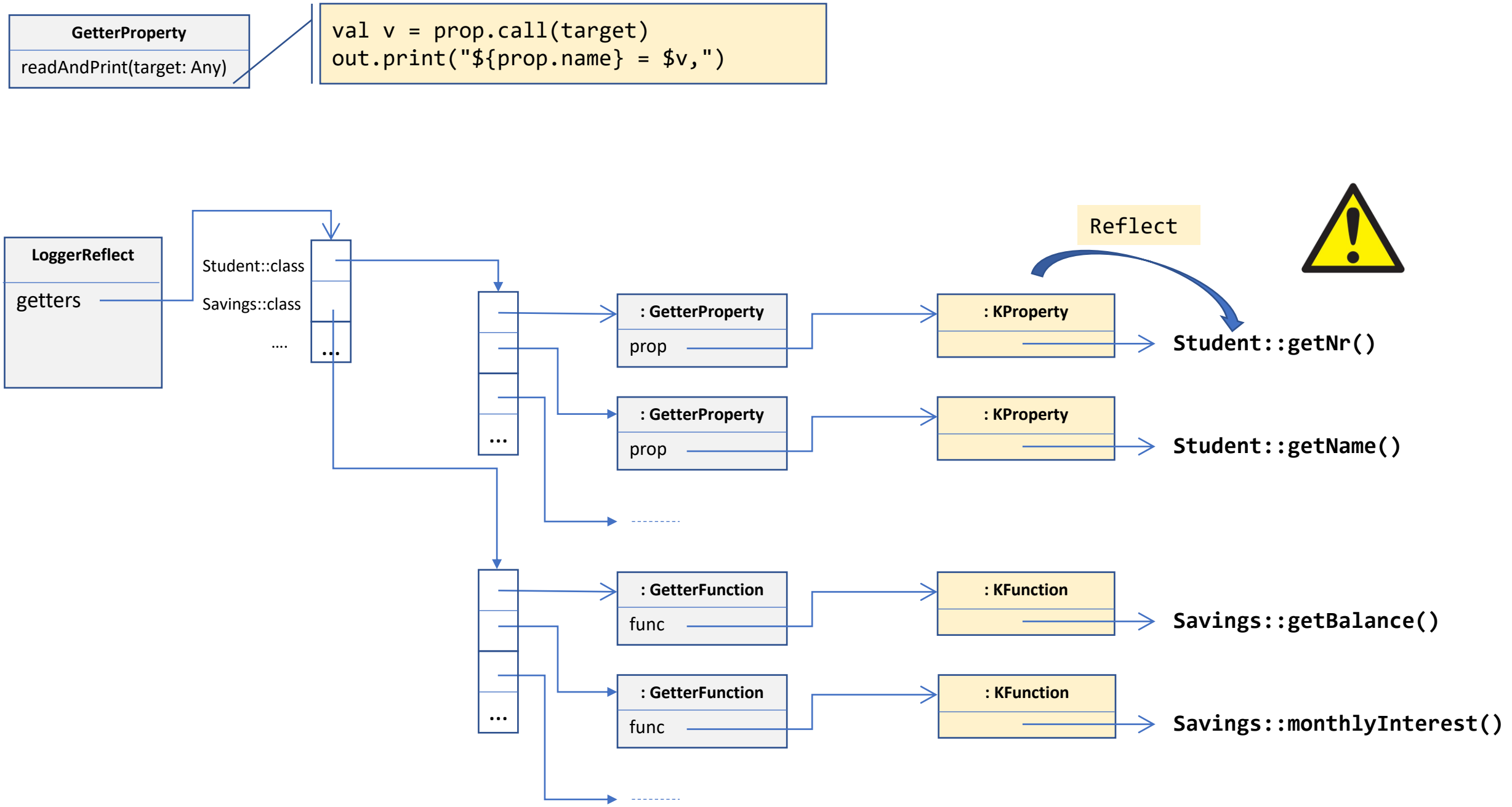
- Remember – Reflection
- Meta-programming e.g. .net Emit
- Remember – Logger via Reflect
- LoggerDynamic

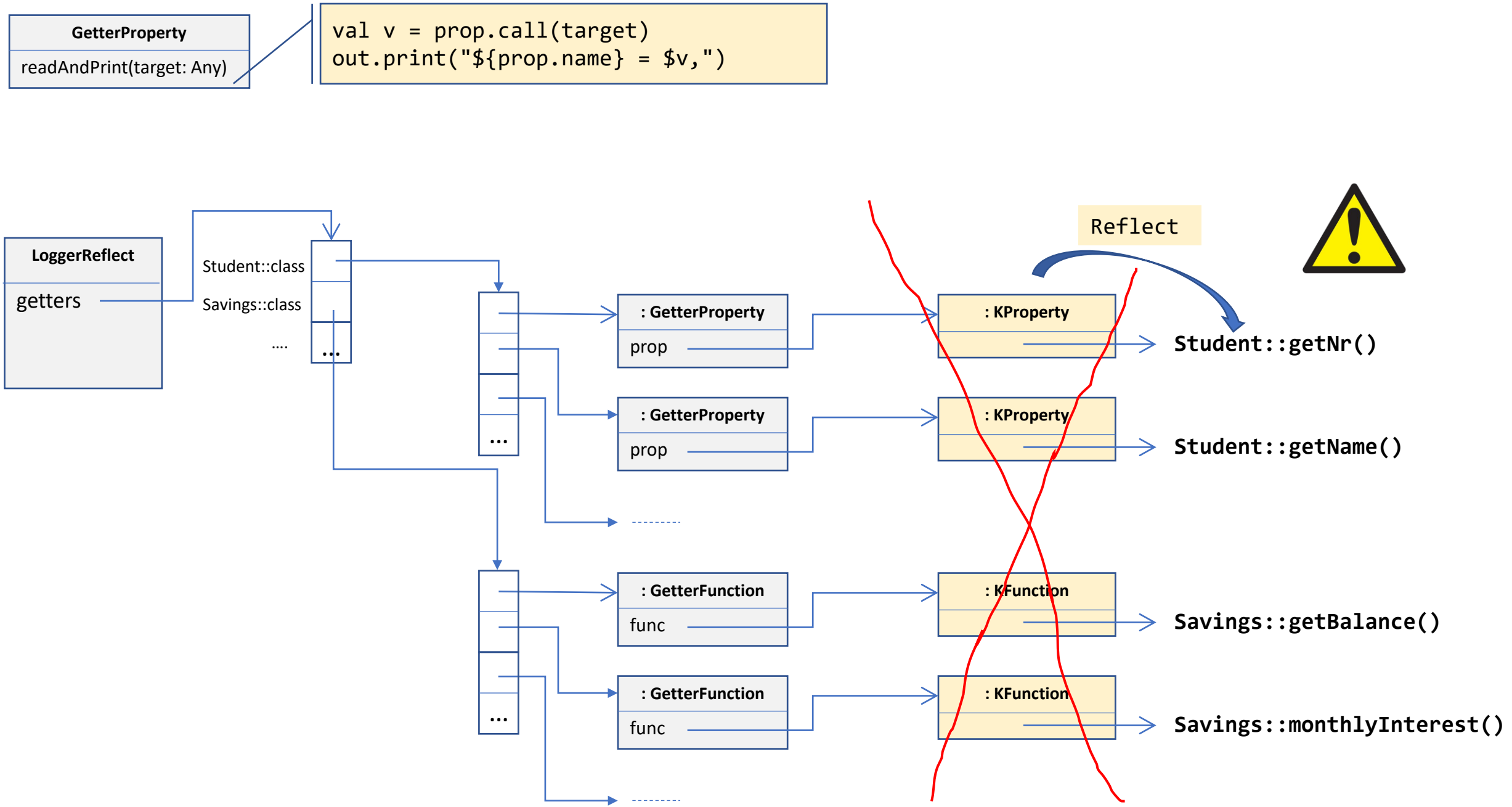
Remember – Logger via Reflect



Remember – Logger via Reflect

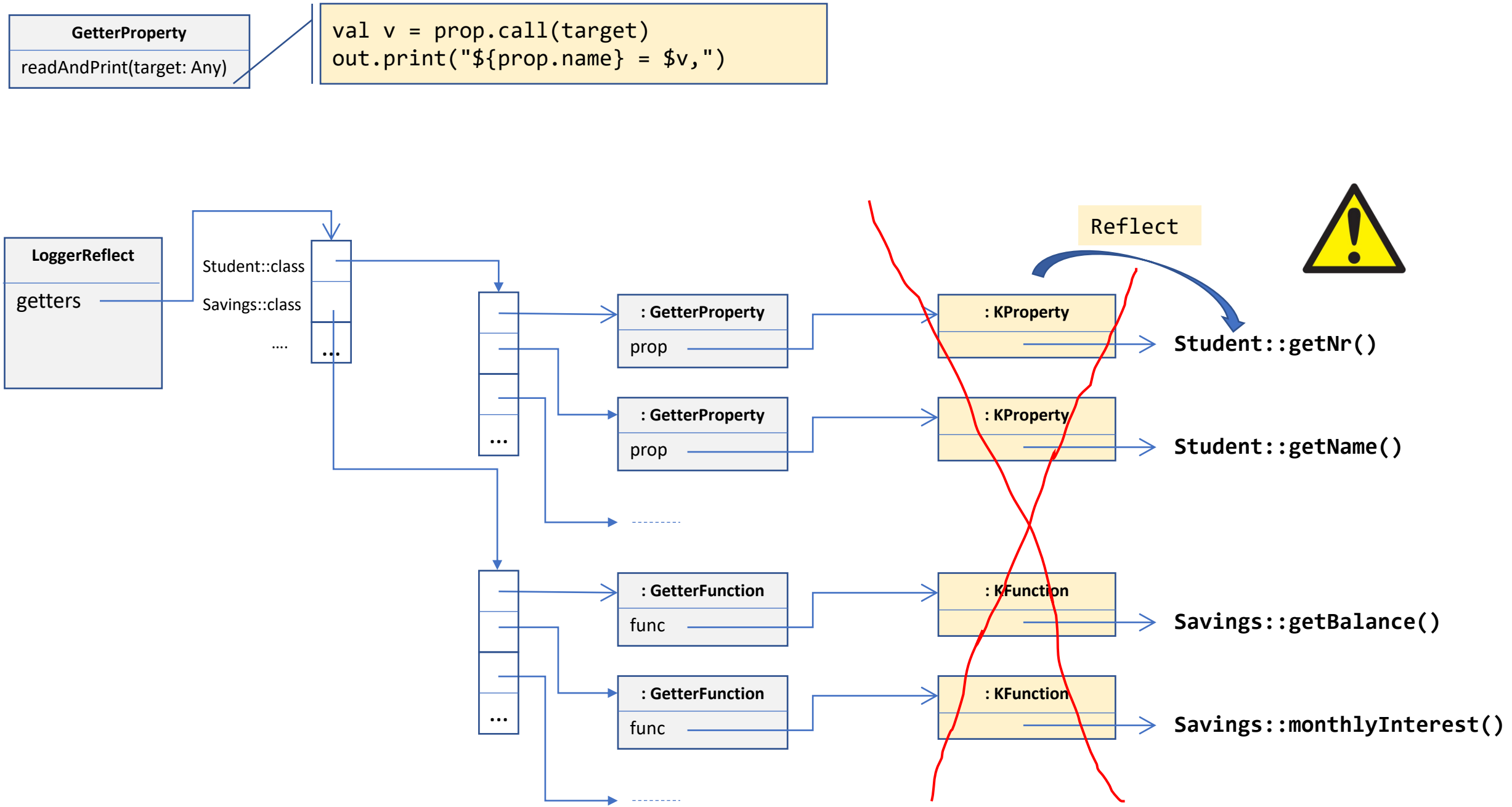


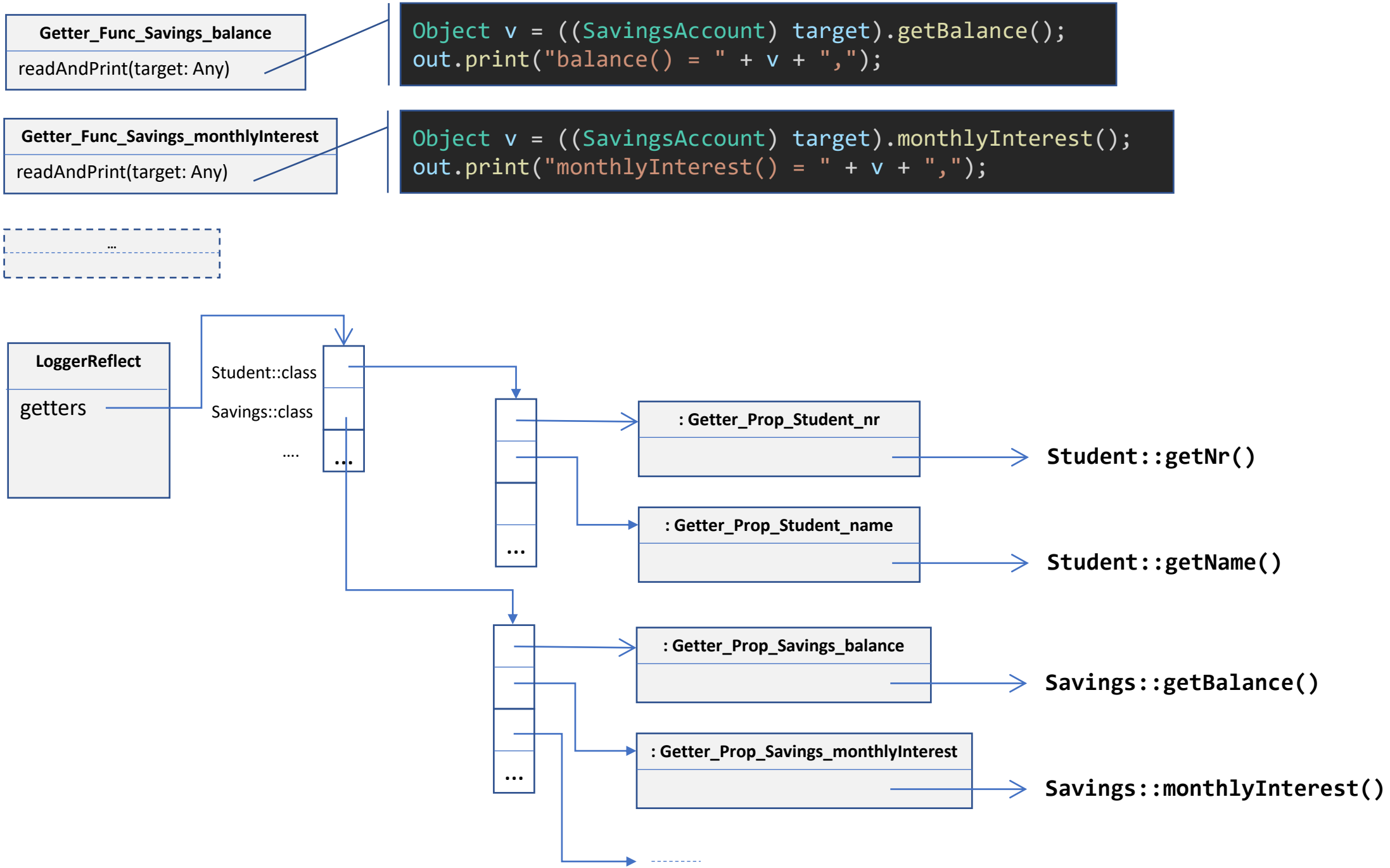

GetterProperty
readAndPrint(target: Any)
val v = prop.call(target)
out.print("${prop.name} = $v,")
LoggerReflect
getters
Student::class
Savings::class
....
...
: GetterProperty
prop
: GetterProperty
prop
: KProperty
: KProperty
Reflect
Student::getNr()
Student::getName()
: GetterFunction
func
: GetterFunction
func
: KFunction
: KFunction
Savings::getBalance()
Savings::monthlyInterest()


GetterProperty
readAndPrint(target: Any)
val v = prop.call(target)
out.print("${prop.name} = $v,")
LoggerReflect
getters
Student::class
Savings::class
....
...
: GetterProperty
prop
: GetterProperty
prop
: GetterFunction
func
: GetterFunction
func
: KProperty
: KProperty
: KFunction
: KFunction
Reflect
Student::getNr()
Student::getName()
Savings::getBalance()
Savings::monthlyInterest()

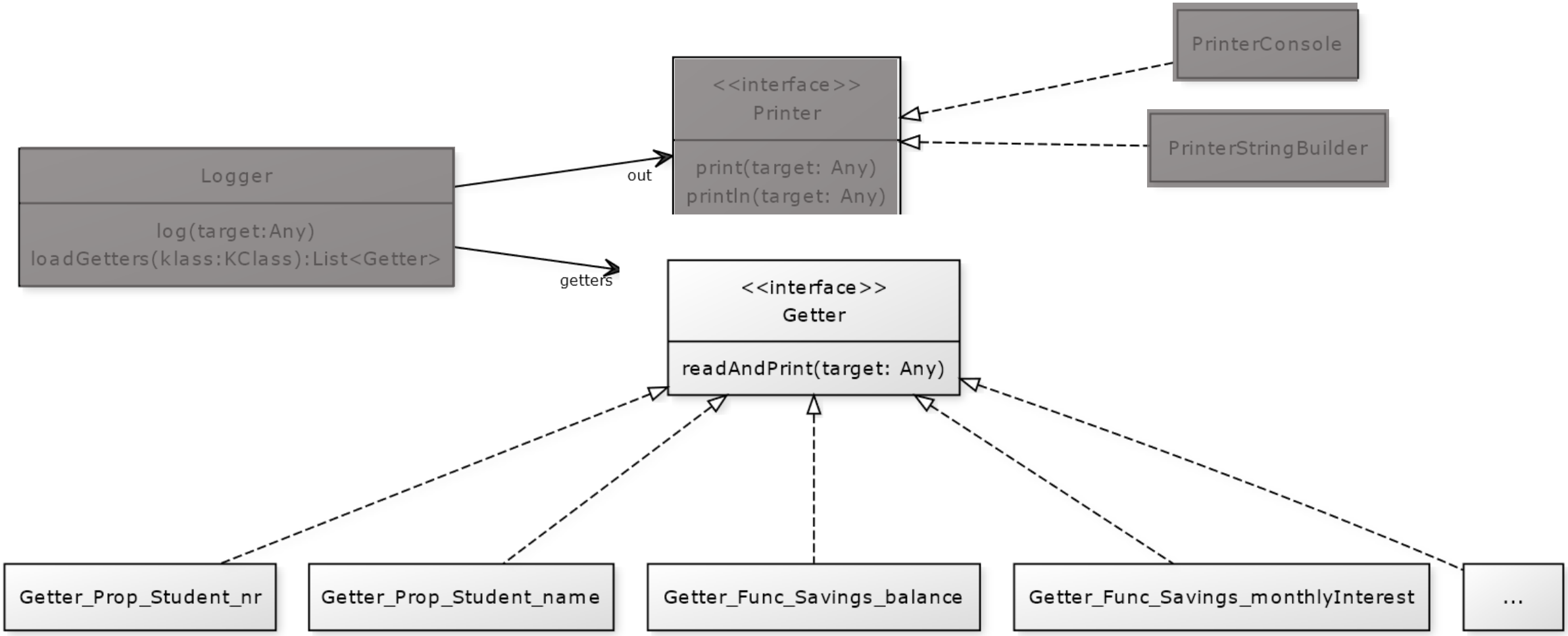
Outline

- Remember – Reflection
- Meta-programming e.g. .net Emit
- Remember – Logger
- LoggerDynamic


GetterProperty
readAndPrint(target: Any)
val v = prop.call(target)
out.print("${prop.name} = $v,")
LoggerReflect
getters
Student::class
Savings::class
....
...
: GetterProperty
prop
: GetterProperty
prop
: GetterFunction
func
: GetterFunction
func
: KProperty
: KProperty
: KFunction
: KFunction
Reflect
Student::getNr()
Student::getName()
Savings::getBalance()
Savings::monthlyInterest()


Getter_Func_Savings_balance
readAndPrint(target: Any)
Object v = ((SavingsAccount) target).getBalance();
out.print("balance() = " + v + ",");
Getter_Func_Savings_monthlyInterest
readAndPrint(target: Any)
Object v = ((SavingsAccount) target).monthlyInterest();
out.print("monthlyInterest() = " + v + ",");
...
LoggerReflect
getters
Student::class
Savings::class
....
...
: Getter_Prop_Student_nr
: Getter_Prop_Student_name
: Getter_Prop_Savings_balance
: Getter_Prop_Savings_monthlyInterest
Student::getNr()
Student::getName()
Savings::getBalance()
Savings::monthlyInterest()

Remember – Logger via Reflect



Remember – Logger via Reflect