Java 核心技术(高阶)

第三章 反射

第二节 反射关键类

华东师范大学 陈良育

反射(1)

- 反射：reflection
  - 程序可以访问、检测和修改它本身状态或行为的能力，即自描述和自控制。
  - 可以在运行时加载、探知和使用编译期间完全未知的类。
  - 给Java插上动态语言特性的翅膀，弥补强类型语言的不足。
  - java.lang.reflect包，在Java 2时代就有，在Java 5得到完善

反射(2)

- 反射：reflection
  - 在运行中分析类的能力
  - 在运行中查看和操作对象
    - 反射构建出无法直接访问的类
    - set或者get到无法访问的成员变量
    - 调用不可访问的方法
  - 实现通用的数组操作代码
  - 类似函数指针的功能

关键类(1)

- Class：类型标识
  - JVM为每个对象都保留其类型标识信息(Runtime Type Identification)
  - 三种获取方式

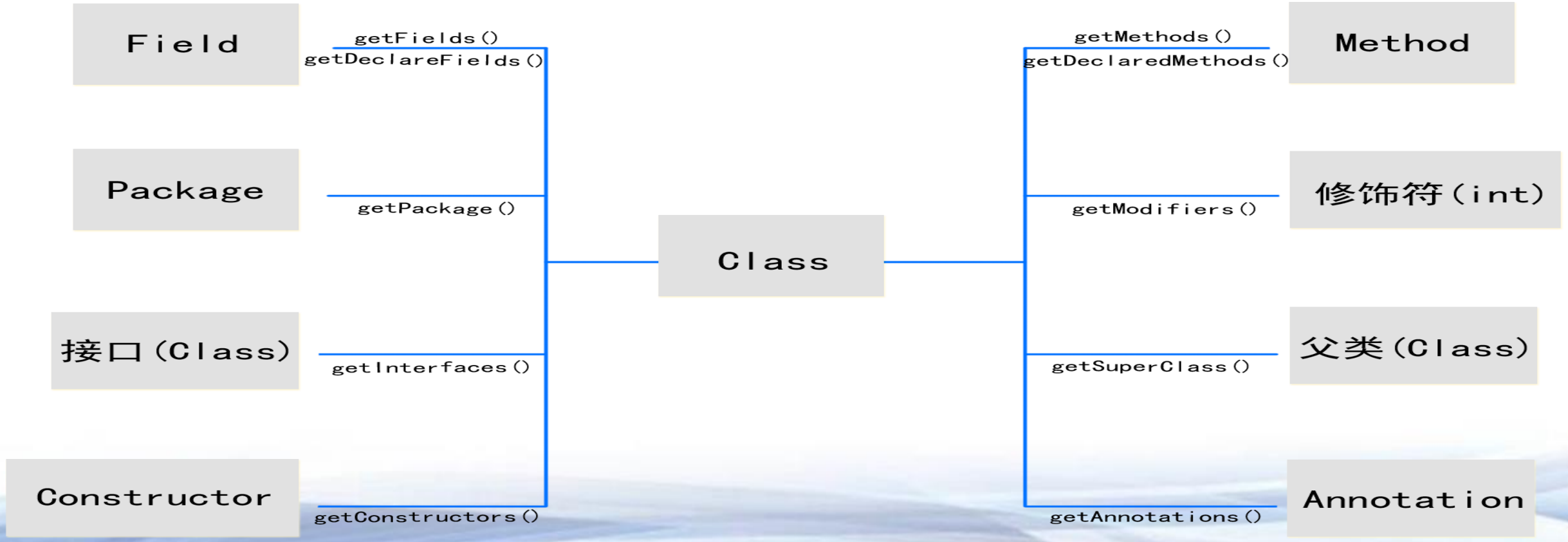
```
String s1 = "abc";
Class c1 = s1.getClass();
System.out.println(c1.getName());

Class c2 = Class.forName("java.lang.String");
System.out.println(c2.getName());

Class c3 = String.class;
System.out.println(c3.getName());
```

关键类(2)

- Class：类型标识
  - 成员变量、方法、构造函数、修饰符、包、父类、父接口……



关键类(3)

- Field：成员变量

```
class A
{
    public int age;
    private String name;

    public A(int age, String name)
    {
        this.age = age;
        this.name = name;
    }
}
```

```
A obj = new A(20, "Tom");
Class c = obj.getClass();

//获取本类及父类所有的public字段
Field[] fs = c.getFields();
System.out.println(fs[0].getName() + ":" + fs[0].get(obj));

//获取本类所有声明的字段
Field[] fs2 = c.getDeclaredFields();
for(Field f : fs2)
{
    f.setAccessible(true);
    System.out.println(f.getName() + ":" + f.get(obj));
}
```

关键类(4)

- Method：成员方法

```
class B {
    public void f1() {
        out.println("B.f1()...");
    }

    private String f2(String s) {
        out.println("B.f2()...");
        return s;
    }
}
```

```
B obj = new B();
Class c = obj.getClass();

// 获取public方法 包括父类和父接口
Method[] ms = c.getMethods();
for (Method m : ms) {
    if ("f1".equals(m.getName())) {
        m.invoke(obj, null);
    }
}

// 获得该类的所有方法
Method[] ms2 = c.getDeclaredMethods();
for (Method m : ms2) {
    if ("f2".equals(m.getName())) {
        m.setAccessible(true);
        String result = (String) m.invoke(obj, "abc");
        out.println(result);
    }
}
```

关键类(5)

- Constructor：构造函数

```
class D
{
    private int num;

    public D() {
        this.num = 10;
    }
    public D(int num) {
        this.num = num;
    }

    public void printNum() {
        System.out.println(this.num);
    }
}
```

```
D d = new D();
Class c = d.getClass();

Constructor[] cons = c.getConstructors();
for (Constructor con : cons) {
    if (con.getParameterCount() > 0) {
        // 有参构造函数
        D obj = (D) con.newInstance(100);
        obj.printNum();
    } else {
        // 无参构造函数
        D obj = (D) con.newInstance();
        obj.printNum();
    }
}
```

关键类(6)

- 父类/父接口

```
class Father { }

class Son extends Father
  implements Cloneable, Comparable
{
    protected Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }

    public int compareTo(Object o) {
        return 0;
    }
}
```

```
Son son = new Son();
Class c = son.getClass();

Class father = c.getSuperclass();
System.out.println(father.getName());

Class[] inters = c.getInterfaces();
for(Class inter : inters)
{
    System.out.println(inter.getName());
}
```

总结

- 了解Java反射的关键类Class
- 根据反射分析类的内容

谢 谢！