**public static void** main(String[] args) {  
 ClassPool pool = ClassPool.getDefault();  
 **boolean** useRuntimeClass = **true**;  
 **if** (useRuntimeClass) {  
 ClassClassPath classPath = **new** ClassClassPath(**new** Rectangle().getClass());  
 pool.insertClassPath(classPath);  
 } **else** {  
 String strClassPath = workDir + **"\\bin"**;  
 pool.insertClassPath(strClassPath);  
 }  
 CtClass cc = pool.get(**"target.Rectangle"**);  
 cc.setSuperclass(pool.get(**"Target.Point"**));  
 cc.writeFile(outputDir);  
  
 CtClass cc = pool.makeClass(**"Point2"**);  
 CtClass cc = pool.makeInterface(**"IPoint"**);  
 ccInterface.writeFile(outputDir);  
  
 CtMethod m = cc.getDeclaredMethod(**"say"**);  
 m.insertBefore(**"{ System.out.println(\"Hello.say:\"); }"**);  
 Class<?> c = cc.toClass();  
 Hello h = (Hello) c.newInstance();  
 h.say();  
  
 Loader cl = **new** Loader(cp);  
 CtClass cc = cp.get(TARGET\_RECTANGLE);  
 cc.setSuperclass(cp.get(TARGET\_POINT));  
 Class<?> c = cl.loadClass(TARGET\_RECTANGLE);  
 Object rect = c.newInstance();  
 System.***out***.println(**"[DBG] rect object: "** + rect);  
 Class<?> rectClass = rect.getClass();  
 Method m = rectClass.getDeclaredMethod(**"getVal"**, **new** Class[] {});  
  
 SampleLoader s = **new** SampleLoader();*//constr: pool = new ClassPool();pool.insertClassPath(inputDir); // MyApp.class must be there.* Class<?> c = s.loadClass(**"MyApp"**);  
 c.getDeclaredMethod(**"main"**, **new** Class[] { String[].**class** }).invoke(**null**, **new** Object[] { args });  
 CtClass cc = pool.get(name);  
 *// modify the CtClass object here* **if** (name.equals(**"MyApp"**)) {  
 CtField f = **new** CtField(CtClass.intType, **"hiddenValue"**, cc);  
 f.setModifiers(Modifier.PUBLIC);  
 cc.addField(f);  
 }  
 **byte**[] b = cc.toBytecode();  
 **return** defineClass(name, b, 0, b.**length**);  
  
 CtMethod m = cc.getDeclaredMethod(**"move"**);  
 m.insertBefore(**"{ System.out.println(\"[DBG] param1: \" + $1); "** + *//* **"System.out.println(\"[DBG] param2: \" + $2); }"**);  
 cc.writeFile(outputDir);  
  
 ClassPool defaultPool = ClassPool.getDefault();  
 defaultPool.insertClassPath(INPUT\_PATH);  
 CtClass cc = defaultPool.get(TARGET\_MYAPP);  
 CtMethod m = cc.getDeclaredMethod(FACT\_METHOD);  
 m.useCflow(FACT\_METHOD);  
 m.insertBefore(**"if ($cflow(fact) == 0)"** + System.*lineSeparator*() + *//* **"System.out.println(\"[MyAppFact Inserted] fact \" + $1);"**);  
 cc.writeFile(OUTPUT\_PATH);  
 InsertMethodBodyCflow s = **new** InsertMethodBodyCflow();*//pool = new ClassPool();pool.insertClassPath(OUTPUT\_PATH); // TARGET must be there.* Class<?> c = s.loadClass(TARGET\_MYAPP);  
 Method mainMethod = c.getDeclaredMethod(**"main"**, **new** Class[] { String[].**class** });  
 mainMethod.invoke(**null**, **new** Object[] { args });  
 *//findClass method:cc = pool.get(name);byte[] b = cc.toBytecode();return defineClass(name, b, 0, b.length);* SubstituteMethodBody s = **new** SubstituteMethodBody();*// pool = new ClassPool();pool.insertClassPath(new ClassClassPath(new java.lang.Object().getClass()));pool.insertClassPath(INPUT\_PATH); // "target" must be there.* Class<?> c = s.loadClass(TARGET\_MY\_APP);  
 Method mainMethod = c.getDeclaredMethod(**"main"**, **new** Class[] { String[].**class** });  
 mainMethod.invoke(**null**, **new** Object[] { args });  
 cc = pool.get(name);  
 cc.instrument(**new** ExprEditor() {  
 **public void** edit(MethodCall m) **throws** CannotCompileException {  
 }  
 **byte**[] **b** = cc.toBytecode();  
 **return** defineClass(name, **b**, 0, **b**.length);  
  
 FieldAcess **s** = **new** FieldAcess();*//pool = new ClassPool();pool.insertClassPath(new ClassClassPath(new java.lang.Object().getClass()));pool.insertClassPath(INPUT\_PATH); // TARGET must be there.* Class<?> **c** = **s**.loadClass(TARGET\_MY\_APP);  
 Method **mainMethod** = **c**.getDeclaredMethod(**"main"**, **new** Class[] { String[].**class** });  
 **mainMethod**.invoke(**null**, **new** Object[] { args });  
  
 NewExprAccess **s** = **new** NewExprAccess();  
 Class<?> **c** = **s**.loadClass(TARGET\_MY\_APP2);  
 Method **mainMethod** = **c**.getDeclaredMethod(**"main"**, **new** Class[] { String[].**class** });  
 **mainMethod**.invoke(**null**, **new** Object[] { args });  
 cc = pool.get(name);  
 cc.instrument(**new** ExprEditor() {  
 **public void** edit(NewExpr newExpr) **throws** CannotCompileException {  
 StringBuilder code = **new** StringBuilder();  
 code.append(**"\"y: \" + "** + **"$\_.y);\n }\n"**);  
 *// System.out.println(code);* newExpr.replace(code.toString());  
  
  
 String src = **"public void xmove(int dx) { x += dx; }"**;  
 CtMethod newMethod = CtNewMethod.make(src, cc);  
 cc.addMethod(newMethod);  
 cc.writeFile(outputDir);  
  
 CtMethod newMethod = CtNewMethod.make(src, cc, **"this"**, **"move"**);  
 cc.addMethod(newMethod);  
 cc.writeFile(outputDir);  
  
 ClassPool pool = ClassPool.getDefault();  
 pool.insertClassPath(inputDir);  
 CtMethod newMethod = **new** CtMethod(CtClass.intType, **"move"**, **new** CtClass[] { CtClass.intType }, cc);  
 cc.addMethod(newMethod);  
 newMethod.setBody(**"{ x += $1; return x;}"**);  
 cc.setModifiers(cc.getModifiers() & ~Modifier.ABSTRACT);  
 cc.writeFile(outputDir);