# Test 1

## Code

|  |
| --- |
| **package** oop1;  **class** A {}  **class** B **extends** A {}  **class** X  {  **int** get(A x) { **return** 0; }  }  **class** Y **extends** X  {  @Override  **int** get(A x) { **return** 1; }  **int** get(B y) { **return** 2; }  }  **public** **class** Test\_1  {  **public** **static** **void** main(String[] args)  {  A a = **new** A();  B b = **new** B();    X xx = **new** X();  Y yy = **new** Y();  X xy = **new** Y();    System.***out***.println(xx.get(a));  System.***out***.println(xx.get(b));  System.***out***.println(yy.get(a));  System.***out***.println(yy.get(b));  System.***out***.println(xy.get(a));  System.***out***.println(xy.get(b));  }  } |

## Resultat

|  |
| --- |
| 0  0  1  2  1  1 |

# Test 2

## Code

|  |
| --- |
| **package** oop1;  **public** **class** Test\_2  {  **public** **static** **void** main(String[] args)  {  Y4 y = **new** Y4();  X4 x = y;  A4 a = **new** B4();  x.foo(**new** B4());  y.foo(**new** B4());  x.foo(a);  y.foo(a);  }  }  **class** A4 {}  **class** B4 **extends** A4 {}  **class** X4  {  **public** **void** foo(A4 a)  {  System.***out***.println("foo(A4)");  };  }  **class** Y4 **extends** X4  {  **public** **void** foo(B4 b)  {  System.***out***.println("foo(B4)");  };  } |

## Resultat

|  |
| --- |
| foo(A4)  foo(B4)  foo(A4)  foo(A4) |

# Test 3

## Code

|  |
| --- |
| **package** oop1;  **public** **class** Test\_3  {  **public** **static** **void** foo(A2 a, A2 b)  {  System.***out***.println(1);  }  **public** **static** **void** foo(A2 a, B2 b)  {  System.***out***.println(2);  }  **public** **static** **void** foo(A2... a)  {  System.***out***.println(3);  }  **public** **static** **void** foo(A2 a, B2... b)  {  System.***out***.println(4);  }  **public** **static** **void** main(String[] args)  {  *foo*(**new** A2(), **new** B2());  }  }  **class** A2 {}  **class** B2 **extends** A2 {} |

## Resultat

|  |
| --- |
| 2 |

# Test 4

## Code

|  |
| --- |
| package ch.fhnw.oop.polymorphism;  public class Test {  public static void main(String[] args) {  C c = new D();  c.f();  }  }  class B {  void f() {  System.out.println("B::f");  h();  }  void h() {  System.out.println("B::h");  }  }  class C extends B {  @Override  void f() {  System.out.println("C::f");  g();  }  void g() {  System.out.println("C::g");  super.f();  }  }  class D extends C {  @Override  void f() {  System.out.println("D::f");  super.f();  }  @Override  void g() {  System.out.println("D::g");  super.g();  }  @Override  void h() {  System.out.println("D::h");  }  } |

## Resultat

|  |
| --- |
| D::f  C::f  D::g  C::g  B::f  D::h |

# Test 5

## Code

|  |
| --- |
| **package** oop1;  **public** **class** Test\_5 {  **public** **static** **void** main(String[] args) {  **new** Y();  }  }  **class** X {  X() {  foo();  }  **private** **void** foo() {  System.***out***.println("X.foo called");  }  }  **class** Y **extends** X {  Y() {  **super**();  foo();  }  **public** **void** foo() {  System.***out***.println("Y.foo called");  }  } |

## Resultat

|  |
| --- |
| X.foo called  Y.foo called |

# Test 6

## Code

|  |
| --- |
| **package** oop1;  **public** **class** Base {  **private** **int** value;  **public** Base() {  value = lookup();  }  **public** **int** lookup() { /\* perform DB lookup \*/  **return** 5;  }  **public** **int** getValue() {  **return** value;  }  **public** **static** **void** main(String[] args) {  Base base = **new** Base();  System.***out***.println(base.getValue());  }  } |

## Resultat

|  |
| --- |
| 5 |

# Test 7

## Code

|  |
| --- |
| **package** oop1;  **class** Derived **extends** Base {  @Override  **public** **int** lookup() { **return** 10; }  }  **public** **class** Check {  **public** **static** **void** main(String[] args) {  Base base = **new** Derived();  System.***out***.println(base.getValue());  }  } |

## Resultat

|  |
| --- |
| 10 |

# Test 8

## Code

|  |
| --- |
| **package** oop1;  **class** Base {  **public** Base(**int** val) {  setup(val);  }  **public** **void** setup(**int** val) {  }  }  **class** Derived **extends** Base {  **private** **int** size = 0;  **public** Derived(**int** val) {  **super**(val);  System.***out***.println("Value constructor: " + val);  }  @Override  **public** **void** setup(**int** val) {  size = val;  System.***out***.println("setup called, size = " + size);  }  **public** **int** getSize() {  System.***out***.println("getSize called, size = " + size);  **return** size;  }  }  **public** **class** Check {  **public** **static** **void** main(String[] args) {  Derived d = **new** Derived(1);  System.***out***.println(d.getSize());  }  } |

## Resultat

|  |
| --- |
| setup called, size = 1  Value constructor: 1  getSize called, size = 0  0 |