COMP2026 Problem Solving Using Object Oriented Programming

# Laboratory 11

**Part A Discovery Exercises**

**Task 1: Upcasting and Downcasting**

1. Create the following classes in an IntelliJ project.

|  |
| --- |
| public class S {  private String name;  public S(String name)  {  this.name = name;  }  public String getName()  {  return name;  }  public void doSomething(){  System.out.println("I am a shape.");  }  } |

|  |
| --- |
| public class C extends S{  private double radius;  public C(String name, double radius)  {  super(name);  this.radius = radius;  }  public double getRadius()  {  return radius;  }  public void doSomething(){  System.out.println("I am a circle.");  }  } |

|  |
| --- |
| public class STester {  public static void main(String[] args) {  new STester().runApp();  }  private void runApp(){  S a = new S("ShapeA");  S b = new C("Circle1", 5);  //upcast the C object to parent type S  }  } |

1. Add the following **doSomething()** method calls in the **STester** class.

|  |
| --- |
| public class STester {  public static void main(String[] args) {  new STester().runApp();  }  private void runApp() {  S a = new S("ShapeA");  S b = new C("Circle1", 5);  a.doSomething();  b.doSomething();  }  } |

1. Run the **STester** program again and paste the output below.

|  |
| --- |
| I am a shape.  I am a circle. |

1. Add the following **getName()** method calls in the **ShapeTester** class.

|  |
| --- |
| public class STester {  public static void main(String[] args) {  new STester().runApp();  }  private void runApp() {  S a = new S("ShapeA");  S b = new C("Circle1", 5);  a.doSomething();  b.doSomething();  System.out.println(a.getName());  System.out.println(b.getName());  }  } |

1. Run the **STester** program again and paste the output below.

|  |
| --- |
| I am a shape.  I am a circle.  ShapeA  Circle1 |

1. Add the following **getRadius()** method call in the **ShapeTester** class.

|  |
| --- |
| public class STester {  public static void main(String[] args) {  new STester().runApp();  }  private void runApp() {  S a = new S("ShapeA");  S b = new C("Circle1", 5);  a.doSomething();  b.doSomething();  System.out.println(a.getName());  System.out.println(b.getName());  System.out.println(b.getRadius());  }  } |

1. Run the **STester** program again and paste the output below. The program should produce errors.

|  |
| --- |
| java: cannot find symbol  symbol: method getRadius()  location: variable b of type S |

1. Downcast **b** to **C** type and call **getRadius()** method again.

|  |
| --- |
| public class STester {  public static void main(String[] args) {  new STester().runApp();  }  private void runApp() {  S a = new S("ShapeA");  S b = new C("Circle1", 5);  a.doSomething();  b.doSomething();  System.out.println(a.getName());  System.out.println(b.getName());  System.out.println(((C) b).getRadius());  }  } |

1. Run the **STester** program again and paste the output below.

|  |
| --- |
| I am a shape.  I am a circle.  ShapeA  Circle1  5.0 |

**Task 2: Understanding Access Modifier**

1. Create an IntelliJ project.
2. Create a package called p1.

Right click the src folder 🡪 New 🡪 Package and then input p1 and click OK.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

1. Right click the package p1 and create a class called Alpha with the following content.

**package** p1;

**public class** Alpha {

**public int** pub;

**protected int** prot;

**int** pack;

**private int** pri;

**public void** methodA() {

System.out.println(this.pub);

System.out.println(this.prot);

System.out.println(this.pack);

System.out.println(this.pri);

}

}

The methodA() tries to print each instance variable. There is no error in the program because the method is in the class Alpha, so it has access to all members of its own class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifier | Public | Protected | No modifier | Private |
| Can access in Alpha? | Y | Y | Y | Y |

1. Create another class called Beta in package p1.

Class Alpha and Beta are inside the same package. Fill in the following table with ‘Y’ for yes and ‘N’ for no.

**package** p1;

**public class** Beta {

**public void** methodB() {

Alpha alphaObj = **new** Alpha();

System.out.println(alphaObj.pub);

System.out.println(alphaObj.prot);

System.out.println(alphaObj.pack);

System.out.println(alphaObj.pri);

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifier | Public | Protected | No modifier | Private |
| Can access in Beta? | Y | Y | Y | N |

1. Create another class called Gamma in package p1.

Gamma is a subclass of class Alpha inside the same package. Fill in the following table with ‘Y’ for yes and ‘N’ for no.

**package** p1;

**public class** Gamma extends Alpha{

**public void** methodG() {

System.out.println(this.pub);

System.out.println(this.prot);

System.out.println(this.pack);

System.out.println(this.pri);

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifier | Public | Protected | No modifier | Private |
| Can access in Gamma? | Y | Y | Y | N |

1. Create another package called p2. Create a class called Delta in package p2.

**package** p2;

**import** p1.\*;

**public class** Delta extends Alpha{

**public void** methodD() {

System.out.println(this.pub);

System.out.println(this.prot);

System.out.println(this.pack);

System.out.println(this.pri);

}

}

Delta is a subclass of class Alpha in a different package. Fill in the following table with ‘Y’ for yes and ‘N’ for no.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifier | Public | Protected | No modifier | Private |
| Can access in Delta? | Y | Y | N | N |

1. Create another class called Epsilon in package p2.

Class Alpha and Epsilon are inside different packages. Fill in the following table with ‘Y’ for yes and ‘N’ for no.

**package** p2;

**import** p1.\*;

**public class** Epsilon{

**public void** methodE() {

Alpha alphaObj = **new** Alpha();

System.out.println(alphaObj.pub);

System.out.println(alphaObj.prot);

System.out.println(alphaObj.pack);

System.out.println(alphaObj.pri);

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifier | Public | Protected | No modifier | Private |
| Can access in Epsilon? | Y | N | N | N |

**References**

1. Bravaco, R., & Simonson, C. (2009). *Java programming: From the ground up*. *Dubuque*, IA: McGraw-Hill.
2. Dean, J., & Dean, R. (2008). *Introduction to programming with Java: A problem solving approach*. Boston: McGraw-Hill.
3. Farrell, J. (2012). *Java programming. Boston, MA: Course Technology Cengage Learning*
4. Forouzan, B. A., & Gilberg, R. F. (2007). *Computer science: A structured programming approach using C (3rd ed.)*. Boston, MA: Thomson Course Technology.
5. Gaddis, T. (2016). *Starting out with Java (6th ed.)*. Pearson.
6. Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8th ed.). Pearson.
7. Schildt, H. ( 2006). *Java a beginner's guide*. New York: McGraw Hill.
8. Schildt, H., & Skrien, D. J. (2013). Java programming: A comprehensive introduction. New York: McGraw-Hill.
9. Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education
10. Xavier, C. (2011). *Java programming: A practical approach*. New Delhi: Tata McGraw Hill.
11. yet another insignificant Programming Notes. (n.d.). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming
12. Zakhour, S., Kannan, S., & Gallardo, R. (2013). *The Java tutorial: A short course on the basics (5th ed.)*.