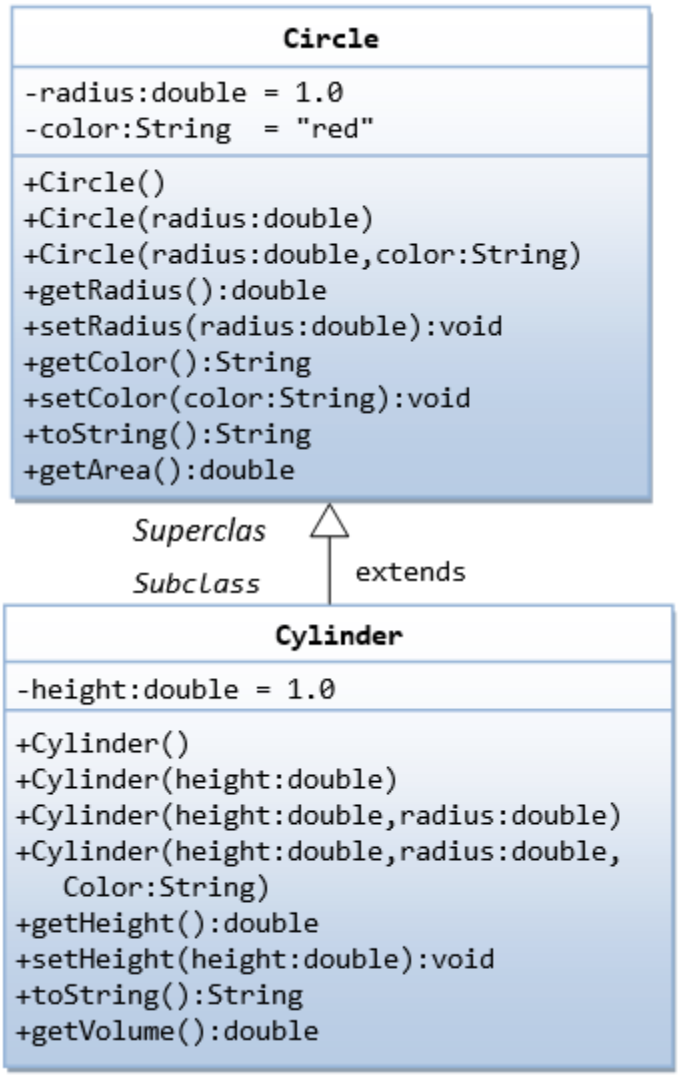
Inheritance

1. Inheritance: Circle and Cylinder Classes



In this example, we derive a subclass called `Cylinder` from the superclass `Circle`, which we have created in the previous chapter. It is important to note that we reuse the class `Circle`. Reusability is one of the most important properties of OOP. (Why reinvent the wheels?) The class `Cylinder` inherits all the member variables (`radius` and `color`) and methods (`getRadius()`, `getArea()`, among others) from its superclass `Circle`. It further defines a variable called `height`, two public methods - `getHeight()` and `getVolume()` and its own constructors, as shown.

- **Circle.Java**

```
 public class Circle {
   // private instance variables
   private double radius;
   private String color;

   // Constructors
   public Circle() {
      this.radius = 1.0;
      this.color = "red";
      System.out.println("Construced a Circle with Circle()");  // for debugging
   }
   public Circle(double radius) {
      this.radius = radius;
      this.color = "red";
      System.out.println("Construced  a  Circle  with  Circle(radius)");    //  for
debugging
   }
   public Circle(double radius, String color) {
      this.radius = radius;
      this.color = color;
      System.out.println("Construced a Circle with Circle(radius, color)");  // for
debugging
   }

   // public getters and setters for the private variables
   public double getRadius() {
      return this.radius;
   }
   public String getColor() {
      return this.color;
   }
   public void setRadius(double radius) {
      this.radius = radius;
   }
   public void setColor(String color) {
      this.color = color;
   }

   /** Returns a self-descriptive String */
   public String toString() {
      return "Circle[radius=" + radius + ",color=" + color + "]";
   }

   /** Returns the area of this Circle */
   public double getArea() {
      return radius * radius * Math.PI;
   }
}
```

- **Cylinder.java**

```
/**
 * A Cylinder is a Circle plus a height.
 */
public class Cylinder extends Circle {
   // private instance variable
   private double height;

   // Constructors
   public Cylinder() {
      super();  // invoke superclass' constructor Circle()
      this.height = 1.0;
      System.out.println("Constructed  a  Cylinder  with  Cylinder()");    //  for
debugging
   }
   public Cylinder(double height) {
      super();  // invoke superclass' constructor Circle()
      this.height = height;
      System.out.println("Constructed a Cylinder with Cylinder(height)");  // for
debugging
   }
   public Cylinder(double height, double radius) {
      super(radius);  // invoke superclass' constructor Circle(radius)
      this.height = height;
      System.out.println("Constructed a Cylinder with Cylinder(height, radius)");
// for debugging
   }
   public Cylinder(double height, double radius, String color) {
      super(radius, color); // invoke superclass' constructor Circle(radius, color)
      this.height = height;
      System.out.println("Constructed  a  Cylinder  with  Cylinder(height,  radius,
color)");  // for debugging
   }

   // Getter and Setter
   public double getHeight() {
      return this.height;
   }
   public void setHeight(double height) {
      this.height = height;
   }

   /** Returns the volume of this Cylinder */
   public double getVolume() {
      return getArea()*height;   // Use Circle's getArea()
   }

   /** Returns a self-descriptive String */
   public String toString() {
```

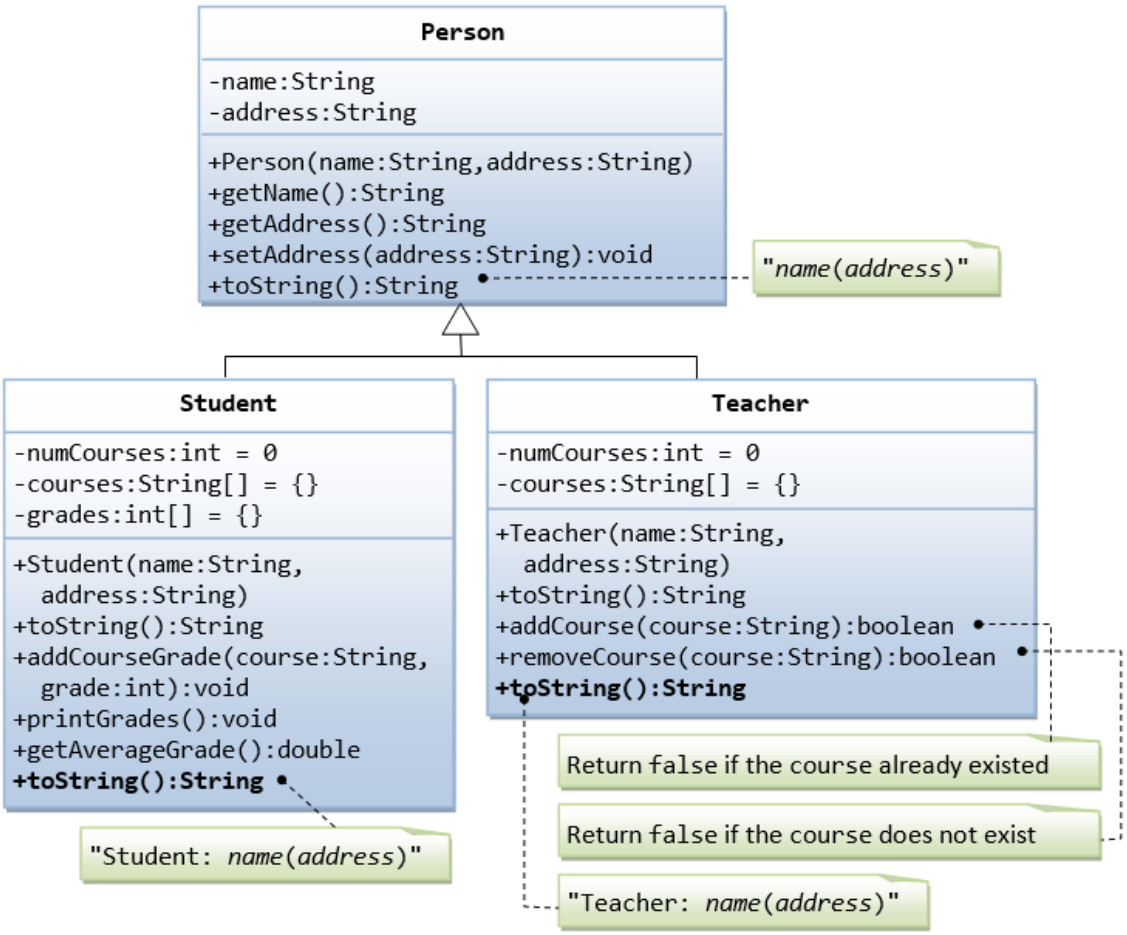
```
      return "This is a Cylinder";  // to be refined later
   }
}
```

- **TestCylinder.Java**

```
/**
 * A test driver for the Cylinder class.
 */
public class TestCylinder {
   public static void main(String[] args) {
      Cylinder cy1 = new Cylinder();
      //Construced a Circle with Circle()
      //Constructed a Cylinder with Cylinder()
      System.out.println("Radius is " + cy1.getRadius()
         + ", Height is " + cy1.getHeight()
         + ", Color is " + cy1.getColor()
         + ", Base area is " + cy1.getArea()
         + ", Volume is " + cy1.getVolume());
      //Radius is 1.0, Height is 1.0, Color is red,
      //Base area is 3.141592653589793, Volume is 3.141592653589793

      Cylinder cy2 = new Cylinder(5.0, 2.0);
      //Construced a Circle with Circle(radius)
      //Constructed a Cylinder with Cylinder(height, radius)
      System.out.println("Radius is " + cy2.getRadius()
         + ", Height is " + cy2.getHeight()
         + ", Color is " + cy2.getColor()
         + ", Base area is " + cy2.getArea()
         + ", Volume is " + cy2.getVolume());
      //Radius is 2.0, Height is 5.0, Color is red,
      //Base area is 12.566370614359172, Volume is 62.83185307179586
   }
}
```

2. Inheritance Superclass Person dan its subclasses



Suppose that we are required to model students and teachers in our application. We can define a superclass called Person to store common properties such as name and address, and subclasses Student and Teacher for their specific properties. For students, we need to maintain the courses taken and their respective grades; add a course with grade, print all courses taken and the average grade. Assume that a student takes no more than 30 courses for the entire program. For teachers, we need to maintain the courses taught currently, and able to add or remove a course taught. Assume that a teacher teaches not more than 5 courses concurrently.

- **The Superclass `Person.java`**

```
/**
 * The superclass Person has name and address.
 */
public class Person {
   // private instance variables
   private String name, address;
```

```
   /** Constructs a Person instance with the given name and address */
   public Person(String name, String address) {
      this.name = name;
      this.address = address;
   }

   // Getters and Setters
   /** Returns the name */
   public String getName() {
      return name;
   }
   /** Returns the address */
   public String getAddress() {
      return address;
   }
   /** Sets the address */
   public void setAddress(String address) {
      this.address = address;
   }

   /** Returns a self-descriptive string */
   @Override
   public String toString() {
      return name + "(" + address + ")";
   }
}
```

- **The Subclass `Student.java`**

```
**
 * The Student class, subclass of Person.
 */
public class Student extends Person {
   // private instance variables
   private int numCourses;   // number of courses taken so far
   private String[] courses; // course codes
   private int[] grades;     // grade for the corresponding course codes
   private static final int MAX_COURSES = 30; // maximum number of courses

   /** Constructs a Student instance with the given name and address */
   public Student(String name, String address) {
      super(name, address);
      numCourses = 0;
      courses = new String[MAX_COURSES];
      grades = new int[MAX_COURSES];
   }

   /** Returns a self-descriptive string */
   @Override
   public String toString() {
```

```
      return "Student: " + super.toString();
   }

   /** Adds a course and its grade - No input validation */
   public void addCourseGrade(String course, int grade) {
      courses[numCourses] = course;
      grades[numCourses] = grade;
      ++numCourses;
   }

   /** Prints all courses taken and their grade */
   public void printGrades() {
      System.out.print(this);
      for (int i = 0; i < numCourses; ++i) {
         System.out.print(" " + courses[i] + ":" + grades[i]);
      }
      System.out.println();
   }

   /** Computes the average grade */
   public double getAverageGrade() {
      int sum = 0;
      for (int i = 0; i < numCourses; i++ ) {
         sum += grades[i];
      }
      return (double)sum/numCourses;
   }
}
```

- **The Subclass `Teacher.java`**

```
/**
 * The Teacher class, subclass of Person.
 */
public class Teacher extends Person {
   // private instance variables
   private int numCourses;   // number of courses taught currently
   private String[] courses; // course codes
   private static final int MAX_COURSES = 5; // maximum courses

   /** Constructs a Teacher instance with the given name and address */
   public Teacher(String name, String address) {
      super(name, address);
      numCourses = 0;
      courses = new String[MAX_COURSES];
   }

   /** Returns a self-descriptive string */
   @Override
   public String toString() {
```

```
      return "Teacher: " + super.toString();
   }

   /** Adds a course. Returns false if the course has already existed */
   public boolean addCourse(String course) {
      // Check if the course already in the course list
      for (int i = 0; i < numCourses; i++) {
         if (courses[i].equals(course)) return false;
      }
      courses[numCourses] = course;
      numCourses++;
      return true;
   }

   /** Removes a course. Returns false if the course cannot be found in the course
list */
   public boolean removeCourse(String course) {
      boolean found = false;
      // Look for the course index
      int courseIndex = -1;  // need to initialize
      for (int i = 0; i < numCourses; i++) {
         if (courses[i].equals(course)) {
            courseIndex = i;
            found = true;
            break;
         }
      }
      if (found) {
         // Remove the course and re-arrange for courses array
         for (int i = courseIndex; i < numCourses-1; i++) {
            courses[i] = courses[i+1];
         }
         numCourses--;
         return true;
      } else {
         return false;
      }
   }
}
```

- **A Test Driver (`TestPerson.java`)**

```
/**
 * A test driver for Person and its subclasses.
 */
public class TestPerson {
   public static void main(String[] args) {
      /* Test Student class */
      Student s1 = new Student("Tan Ah Teck", "1 Happy Ave");
```

```
        s1.addCourseGrade("IM101", 97);
        s1.addCourseGrade("IM102", 68);
        s1.printGrades();
        //Student: Tan Ah Teck(1 Happy Ave) IM101:97 IM102:68
        System.out.println("Average is " + s1.getAverageGrade());
        //Average is 82.5

        /* Test Teacher class */
        Teacher t1 = new Teacher("Paul Tan", "8 sunset way");
        System.out.println(t1);
        //Teacher: Paul Tan(8 sunset way)
        String[] courses = {"IM101", "IM102", "IM101"};
        for (String course: courses) {
           if (t1.addCourse(course)) {
              System.out.println(course + " added");
           } else {
              System.out.println(course + " cannot be added");
           }
        }
        //IM101 added
        //IM102 added
        //IM101 cannot be added
        for (String course: courses) {
           if (t1.removeCourse(course)) {
              System.out.println(course + " removed");
           } else {
              System.out.println(course + " cannot be removed");
           }
        }
        //IM101 removed
        //IM102 removed
        //IM101 cannot be removed
    }
}
```