**Lab 12 30-10-2017**

**Object Oriented Concepts and Programming**

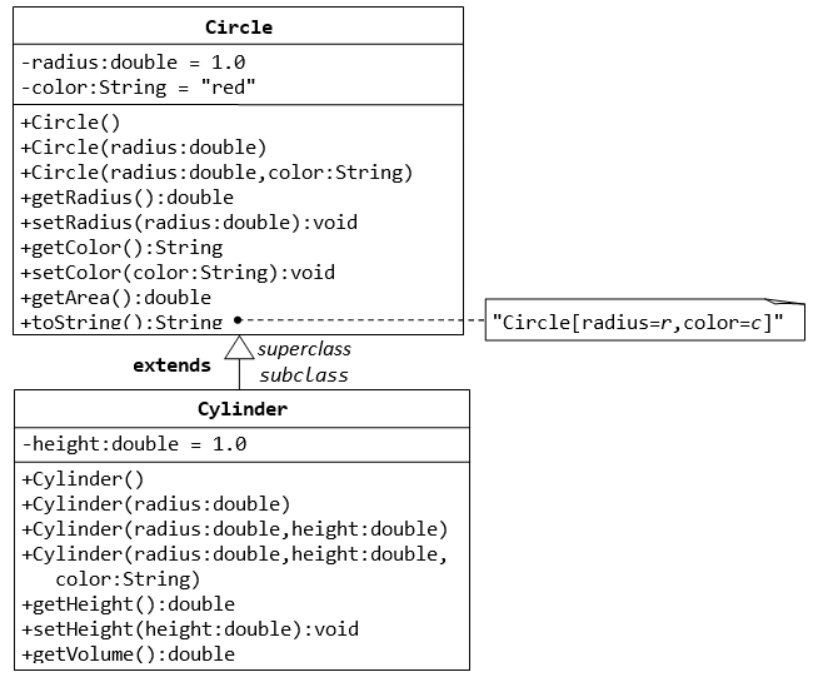
**Learning Objectives**

1. The students will be able to

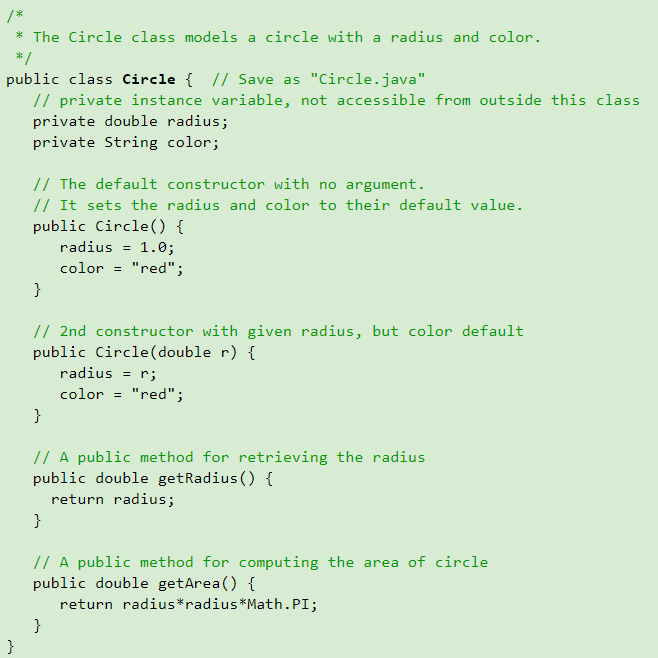
* Implement the inheritance relationship between two classes having an Is-A relationship
* Override and reuse base class behavior in the derived class
* Override the common object class methods such as toString
* Observe and use the super keyword for accessing members/constructors of the super-class

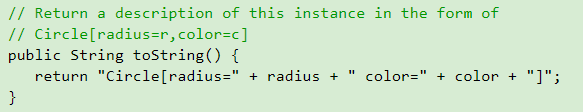
|  |
| --- |
| **Lab Walkthrough/Demo** |

**Lab Demo 12.1.** Inheritance in action. **Reference**: <https://www3.ntu.edu.sg/home/ehchua/programming/java/j3f_oopexercises.html>

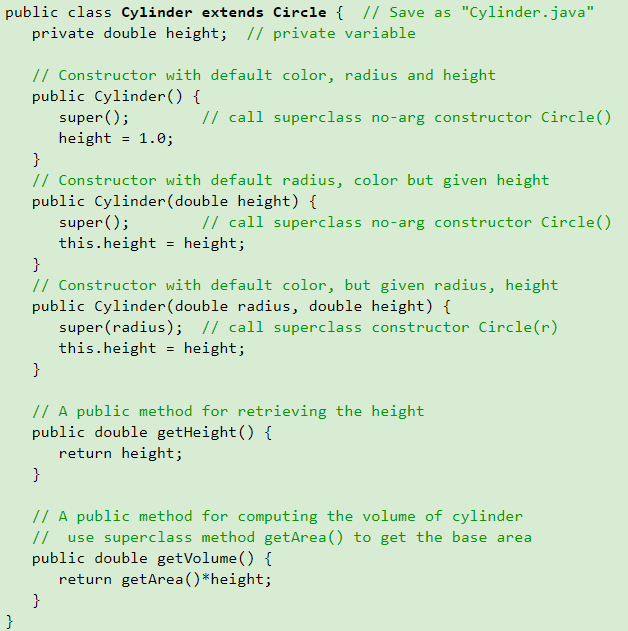


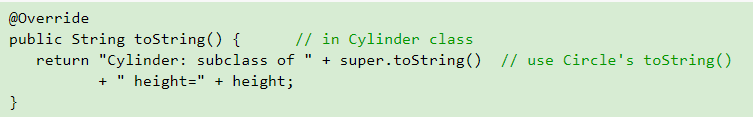
**THE Circle CLASS**





**THE Cylinder CLASS**





**THE Test CLASS**

public class **TestCylinder** { // save as "TestCylinder.java"

public static void main (String[] args) {

// Declare and allocate a new instance of cylinder

// with default color, radius, and height

Cylinder c1 = new Cylinder();

// Declare and allocate a new instance of cylinder

// specifying height, with default color and radius

Cylinder c2 = new Cylinder(10.0);

// Declare and allocate a new instance of cylinder

// specifying radius and height, with default color

Cylinder c3 = new Cylinder(2.0, 10.0);

System.out.println("Cylinder1 data:" + c1

+ " Cylinder2 data=" + c2

+ " Cylinder3 data=" + c3);

}

}

**What will be the output?**

|  |
| --- |
| **Lab Task** |

Lab Task **12.1**

The Source Class:

Now complete the following tasks inside the **Employee** class:

1. Consider a base class named **Employee** and its derived classes **HourlyEmployee** and **PermanentEmployee** while taking into account the following criteria.
   1. Employee class has two data fields i.e. a name (of type string) and specific empID (of type integer)
   2. It has a toString method, which returns a string in the following format.
      1. Employee: Ali (ID: 786)
   3. Both classes (HourlyEmployee and PermanentEmployee) have an attribute named **hourlyIncome**, while the HourlyEmployee has an extra field named totalHours
   4. Both classes (HourlyEmployee and PermanentEmployee) have three-argument constructors to initialize the hourlyIncome as well as data fields of the base class.
      1. The id and name should be initialized using the constructor of the employee class
      2. The totalHours for HourlyEmployee should be input from user right inside the constructor
   5. Class HourlyEmployee has a function named calculate\_hourly\_income to calculate the income of an employee for the actual number of hours he or she worked.
   6. Similarly, PermanentEmployee class has function named calculate\_income to calculate the income of an employee that gets paid the salary for exact 180 hours, no matter how many actual hours he or she worked.
   7. Implement the toString Method for both classes. The method should return the string formatted according to following for each class:

|  |  |
| --- | --- |
| HourlyEmployee | PermanentEmployee |
| Employee: Ali (ID: 786)  This month salary = 150 \* 500 = 75000 | Employee: Ali (ID: 786)  This month salary @500 PKR/Hour = 90000 |

Printed using toString of Employee class

The Test Class:

1. Develop the main() method inside the employee class, create an instance of both classes (i.e. HourlyEmployee and PermanentEmployee) with the data of your own choice.
2. Test the working of functions that calculate the total income of an employee.
3. Test the toString methods of both classes by printing both objects in a print statement.

|  |
| --- |
| **Home Tasks** |

1. Go to following url:

<https://www3.ntu.edu.sg/home/ehchua/programming/java/j3f_oopexercises.html>

Solve exercises 4.2, 4.3, 4.4 and 4.5. For more information on the classes involved in these exercises, you may scroll up the page and find your desired class. Feel free to discuss if there is any difficulty.

1. Consider a class BankAccount that has

* two attributes i.e. accountID and balance and
* A function named balanceInquiry() to get information about the current amount in the account
* Derive two classes from the BankAccount class i.e. CurrentAccount and the SavingsAccount. Both classes (CurrentAccount and SavingsAccount) inherit all attributes/behaviors from the BankAccount class. In addition, followings are required to be the part of both classes
  + Appropriate constructors to initialize data fields of base class
  + A function named amountWithdraw(amount) to withdraw certain amount while taken into account the following conditions
    - While withdrawing from current account, the minimum balance should not decrease Rs. 5000
    - While withdrawing from savings account, the minimum balance should not decrease Rs. 10,000

1. amountDeposit(amount) to deposit amount in the account

* In the main() function, create instances of derived classes (i.e. CurrentAccount and SavingsAccount) and invoke their respective functions to test their working.