

**Department of Computing**

**CS 212: Object Oriented Programming**

**Class: BESE-11 AB**

**Lab 08: Classes & Objects II**

**Date: May 28th, 2021**

**Instructor:**

**Hania Aslam**

CS 212: Object Oriented Programming Page 1



**Learning Objectives**

The learning objective of this lab is to understand and practice the concepts of constructors, constructor chaining, copy constructor, this reference and overloaded methods.

**Activity #1**

Complete the following Time Class and provide the missing constructor definitions. For each constructor, you need to use the **setTime** method for setting-up the parameters that are passed to it. In this way, you can ensure that all parameters are valid. Also, see what happens if you pass an invalid parameter to the constructor, while creating an object of Time class (as done for t4 object).

class Time

{

private int hour; // 0 - 23

private int minute; // 0 - 59

private int second; // 0 - 59

// TODO: Add constructors definitions here

// set a new time value using universal time; throw an

// exception if the hour, minute or second is invalid

public void setTime( int h, int m, int s )

{

// validate hour, minute and second

if ( ( h >= 0 && h < 24 ) && ( m >= 0 && m < 60 ) && ( s >= 0 && s < 60 ) )

{

hour = h; minute = m; second = s;

} // end if else

throw new IllegalArgumentException("hour, minute and/or second was out of range" );

} // end method setTime

// convert to String in universal-time format (HH:MM:SS)

public String toUniversalString()

{

return String.format( "%02d:%02d:%02d", hour, minute, second );

} // end method toUniversalString

public String toString()

{

return String.format( "%d:%02d:%02d %s",

( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ), minute, second, ( hour < 12 ? "AM" : "PM" ) );

} // end method toString

} // end class Time

// This class tests the Time Class by creating different objects.

public class TimeTest{

public static void main(String [] args){

Time t0 = new Time(); // Set Time to 00:00:00

Time t1 = new Time(11); // Set Time to 11:00:00

Time t2 = new Time(12, 40);// Set Time to 12:40:00

Time t3 = new Time(23, 40, 55); // Set Time to 23:40:55

Time t4 = new Time(23, 40, 65); // Set Time to 23:40:65

// Print All Times in Universal Format

System.out.println(t0.toUniversalString());

System.out.println(t1.toUniversalString());

System.out.println(t2.toUniversalString());

System.out.println(t3.toUniversalString());

// Print All Times in Standard Format

System.out.println(t0);// toString method invoked on objects implicitly

System.out.println(t1);

System.out.println(t2);

System.out.println(t3);

}

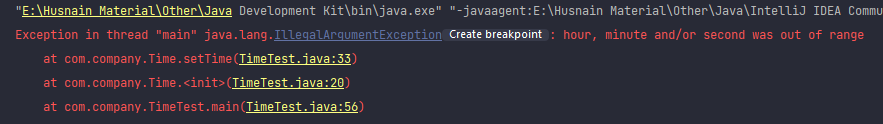
}

**Solution:**

The constructors for the above code are defined as follows:

Time(){  
 setTime(0,0,0);  
}  
Time(int *hour*){  
 setTime(*hour*,0,0);  
}  
Time(int *hour*,int *minute*){  
 setTime(*hour*,*minute*,0);  
}  
Time(int *hour*,int *minute*,int *second*){  
 setTime(*hour*,*minute*,*second*);  
}

Now if we make a new object of the class and pass an invalid parameter, it will throw the following error:

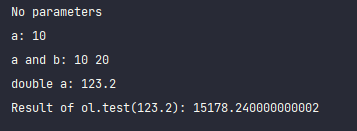


**Activity #2**

What will be the output of the following program? Execute and confirm your understanding.

|  |
| --- |
| // Demonstrate method overloading.  class Overload {  void test() {  System.out.println("No parameters");  }  // Overload test for one integer parameter.  void test(int a) {  System.out.println("a: " + a);  }  // Overload test for two integer parameters.  void test(int a, int b) {  System.out.println("a and b: " + a + " " + b);  }  // Overload test for a double parameter  double test(double a) {  System.out.println("double a: " + a);  return a\*a;  }  }  public class OverloadTest {  public static void main(String args[]) {  Overload ol = new Overload();  double result;  // call all versions of test()  ol.test();  ol.test(10);  ol.test(10, 20);  result = ol.test(123.2);  System.out.println("Result of ol.test(123.2): " + result);  }  } |

**Output:**

****

**Activity #3**

Execute the following code snippets one by one and rectify and explain the errors in each of them:

public class Circle {

private double radius;

public Circle(double radius){

radius=radius;

}

public Circle(){

this(10.0);

}

public void display(){

System.out.println(radius);

}

public static void main(String[] args) {

Circle c1=new Circle();

c1.display();

}

}

public class Circle {

private double radius;

public Circle(double radius){

this.radius=radius;

}

public Circle(){

System.out.println("Calling parameterized constructor next...");

this(10.0);

}

public void display(){

System.out.println(radius);

}

public static void main(String[] args) {

Circle c1=new Circle();

c1.display();

}

}

public class Circle {

private double radius;

public Circle(double radius){

this(10.0);

this.radius=radius;

}

public Circle(){

}

public void display(){

System.out.println(radius);

}

public static void main(String[] args) {

Circle c1=new Circle();

c1.display();

}

}

public class Circle {

private double radius;

public Circle(double radius){

this();

this.radius=radius;

}

public Circle(){

this(10.0);

}

public void display(){

System.out.println(radius);

}

public static void main(String[] args) {

Circle c1=new Circle();

c1.display();

}

}

**Snippet 1:**

The error produced in this program is a logical error i.e., it does not print the required value of radius which is 10.0 instead it prints 0.0. This is because the compiler cannot differentiate between the parameter ‘radius’ and the field ‘radius’ on **line 4** because both have same name. So, it runs the program with the garbage value of radius which is 0.0. This error can be rectified by using **‘this’** keyword before the field so that it refers to the field of the class.

**Snippet 2:**

The error produced in this program is due to the recursive statement ‘this(10.0)’ present in the parameterized constructor as the constructor is invoked infinitely and the program crashes. To rectify this error we just have to move the ‘this(10.0)’ statement into the no-arg constructor ‘Circle()’ for the program to run fine.

**Snippet 3:**

When applying constructor chaining, there are some rules which we need to keep in mind. Among one of these rules is that the constructor invoking statement ‘this(parameters….)’ must be the first statement in the constructor from where it is being called. As in our program, the print statement comes first, hence it gives the error. To rectify the error, we just have to have to write the ‘this’ statement prior to the print statement.

**Snippet 4:**The error in this code is due to recursion occurring between the no-arg constructor ‘Circle()’ and the parameterized constructor ‘Circle(double radius)’ via the statements ‘this()’ and ‘this(10.0)’. The error in the code can be rectified by **removing the ‘this()’ statement** from the parameterized constructor ‘Circle(double radius)’.

**Lab Task#1:**

Define a class BMI.java that computes body mass index according to the following class diagram:

|  |
| --- |
| BMI |
| -name : String  -age : int  -weight : double  -height : double |
| +BMI(name : String, age : int, weight : double, height : double)  +BMI(name : String, weight : double, height : double)  +getBMI():double  +getStatus():String  +getAge() : int  +getName(): String  +getWeight(): double  +getHeight(): double  +setName(name: String):void  +setAge(age: int):void  +setWeight(weight: double):void  +setHeight(height: double):void |

Your program shall prompt the user to enter weight in pounds and height in inches. An individual’s BMI shall be calculated inside method getBMI() and method getStatus() must return the appropriate results i.e: Underweight, Normal e.t.c. Next, add a copy constructor to the class BMI.java and demonstrate the correct working of your BMI.java class using a test program.

**Note:** Use the formula, conversions and status ranges provided below to compute user’s BMI results.

**BMI Status:**

Less than 18.5 => Underweight

Between 18.5 to 24.9 => Normal

Between 25 to 29.9 => Overweight

30 or more than 30 => Obese

BMI Formula = weight \* KG\_PER\_POUND / (height \* METERS\_PER\_INCH \*height \* METERS\_PER\_INCH)

KG\_PER\_POUND=0.45359237

METERS\_PER\_INCH=0.0254

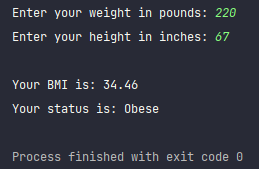
**Code:**

package com.company;  
  
*//Creating our own class BMI*public class BMI {  
 *//Declaring fields* String name;  
 int age;  
 double weight;  
 double height;  
  
 *//Defining the no-arg constructor* BMI(){  
 }  
 *//Defining the 4-parameter parameterized constructor* BMI(String *name*, int *age*, double *weight*, double *height*){  
 this.name = *name*;  
 this.age = *age*;  
 this.weight = *weight*;  
 this.height = *height*;  
 }  
 *//Defining the triple parameterized constructor* BMI(String *name*, double *weight*, double *height*){  
 this.name = *name*;  
 this.weight = *weight*;  
 this.height = *height*;  
 }  
 *//Defining the copy constructor* BMI(BMI *person*){  
 this.name = *person*.name;  
 this.age = *person*.age;  
 this.weight = *person*.weight;  
 this.height = *person*.height;  
 }  
 *//Name setter method* public void setName(String *name*){this.name = *name*;}  
 *//Age setter method* public void setAge(int *age*){this.age = *age*;}  
 *//Weight setter method* public void setWeight(double *weight*) {this.weight = *weight*;}  
 *//Height setter method* public void setHeight(double *height*){this.height = *height*;}  
  
 *//Age getter method* public int getAge(){  
 return age;  
 }  
 *//Name getter method* public String getName(){  
 return name;  
 }  
 *//Weight getter method* public double getWeight(){  
 return weight;  
 }  
 *//Height getter method* public double getHeight(){  
 return height;  
 }  
  
 *//Method for calculating BMI* public double getBMI(){  
 double weight\_in\_kg = weight \* 0.4536;  
 double height\_in\_meters = height \* 0.0254;  
 return weight\_in\_kg / (height\_in\_meters \* height\_in\_meters);  
 }  
  
 *//Method for calculating the status of person* public String getStatus(double *bmi*){  
 String status = "";  
 if (*bmi* < 18.5){  
 status = "Underweight";  
 }  
 else if (*bmi* >= 18.5 && *bmi* <= 24.9){  
 status = "Normal";  
 }  
 else if (*bmi* >= 25 && *bmi* <= 29.9){  
 status = "Overweight";  
 }  
 else if (*bmi* >= 30){  
 status = "Obese";  
 }  
 return status;  
 }  
  
}

**Test Program:**

package com.company;  
  
import java.util.Scanner;  
  
*//Creating the test class BMITest*public class BMITest {  
  
 *//Main method* public static void main(String[] *args*) {  
  
 *//Creating an object of Scanner class to take input* Scanner sc = new Scanner(System.*in*);  
  
 *//Taking inputs* System.*out*.print("Enter your weight in pounds: ");  
 double weight = sc.nextDouble();  
 System.*out*.print("Enter your height in inches: ");  
 double height = sc.nextDouble();  
  
 *//Creating an object of BMI class* BMI person = new BMI();  
 *//Setting weight of person* person.setWeight(weight);  
 *//Setting height of person* person.setHeight(height);  
  
 *//Getting BMI of person* double person\_bmi = person.getBMI();  
 System.*out*.println();  
 *//Printing BMI* System.*out*.printf("Your BMI is: %.2f\n", person\_bmi);  
 *//Printing status of person* System.*out*.println("Your status is: " + person.getStatus(person\_bmi));  
 }  
}

**Output Screenshot:**

****

**Lab Task#2:**

Modify class Time2 of (as given below) to include a tick method that increments the time stored in a Time2 object by one second. Provide method incrementMinute to increment the minute by one and method incrementHour to increment the hour by one. Write a program that tests the tick method, the incrementMinute method and the incrementHour method to ensure that they work correctly. Be sure to test the following cases:

a) incrementing into the next minute and

b) incrementing into the next hour

c) incrementing into the next day (i.e., 11:59:59 PM to 12:00:00 AM).

**Note: Please read comments to understand the code snippet.**

public class Time2

{

private int hour; // 0 - 23

private int minute; // 0 - 59

private int second; // 0 - 59

// Time2 no-argument constructor:

// initializes each instance variable to zero

public Time2() {

this( 0, 0, 0 ); // invoke Time2 constructor with three arguments

} // end Time2 no-argument constructor

//Time2 constructor: hour supplied, minute and second defaulted to 0

public Time2( int h ){

this( h, 0, 0 ); // invoke Time2 constructor with three arguments

} // end Time2 one-argument constructor

//Time2 constructor: hour and minute supplied, second defaulted to 0

public Time2( int h, int m ) {

this( h, m, 0 ); // invoke Time2 constructor with three arguments

} // end Time2 two-argument constructor

//Time2 constructor: hour, minute and second supplied

public Time2( int h, int m, int s ) {

setTime( h, m, s ); // invoke setTime to validate time

} // end Time2 three-argument constructor

//Time2 constructor: another Time2 object supplied

public Time2( Time2 time )

{

// invoke Time2 three-argument constructor

this( time.getHour(), time.getMinute(), time.getSecond() );

} // end Time2 constructor with a Time2 object argument

// Set Methods

// set a new time value using universal time;

// validate the data

public void setTime( int h, int m, int s )

{

setHour( h ); // set the hour

setMinute( m ); // set the minute

setSecond( s ); // set the second

} // end method setTime

//validate and set hour

public void setHour( int h ){

if ( h >= 0 && h < 24 ) hour = h;

else

throw new IllegalArgumentException( "hour must be 0-23" );

} // end method setHour

//validate and set minute

public void setMinute( int m ){

if ( m >= 0 && m < 60 )

minute = m;

else

throw new IllegalArgumentException( "minute must be 0-59" );

} // end method setMinute

//validate and set second

public void setSecond( int s ) {

if ( s >= 0 && s < 60 )

second = ( ( s >= 0 && s < 60 ) ? s : 0 );

else

throw new IllegalArgumentException( "second must be 0-59" );

} // end method setSecond

// Get Methods

public int getHour(){

return hour;

}

public int getMinute(){

return minute;

}

public int getSecond(){

return second;

}

//convert to String in universal-time format (HH:MM:SS)

public String toUniversalString(){

return String.format("%02d:%02d:%02d", getHour(), getMinute(), getSecond() );

}

//convert to String in standard-time format (H:MM:SS AM or PM)

public String toString(){

return String.format( "%d:%02d:%02d %s",

( (getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12 ),

getMinute(), getSecond(), ( getHour() < 12 ? "AM" : "PM" ) );

} // end method toString

} // end class Time2

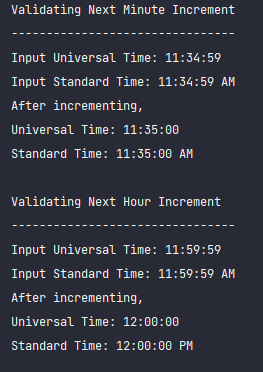
**Code:**

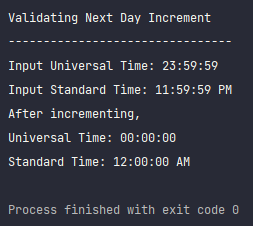
package com.company;  
  
*//Creating our own class Time2*public class Time2  
{  
 *//Declaring fields* private int hour; *// 0 - 23* private int minute; *// 0 - 59* private int second; *// 0 - 59  
  
 // Time2 no-argument constructor:  
// initializes each instance variable to zero* public Time2() {  
 this( 0, 0, 0 ); *// invoke Time2 constructor with three arguments* } *// end Time2 no-argument constructor  
  
 //Time2 constructor: hour supplied, minute and second defaulted to 0* public Time2( int *h* ){  
  
 this( *h*, 0, 0 ); *// invoke Time2 constructor with three arguments* } *// end Time2 one-argument constructor  
  
  
 //Time2 constructor: hour and minute supplied, second defaulted to 0* public Time2( int *h*, int *m* ) {  
 this( *h*, *m*, 0 ); *// invoke Time2 constructor with three arguments* } *// end Time2 two-argument constructor  
  
 //Time2 constructor: hour, minute and second supplied* public Time2( int *h*, int *m*, int *s* ) {  
 setTime( *h*, *m*, *s* ); *// invoke setTime to validate time* } *// end Time2 three-argument constructor  
  
 //Time2 constructor: another Time2 object supplied* public Time2( Time2 *time* )  
 {  
*// invoke Time2 three-argument constructor* this( *time*.getHour(), *time*.getMinute(), *time*.getSecond() );  
 } *// end Time2 constructor with a Time2 object argument  
// Set Methods  
// set a new time value using universal time;  
// validate the data* public void setTime( int *h*, int *m*, int *s* )  
 {  
 setHour( *h* ); *// set the hour* setMinute( *m* ); *// set the minute* setSecond( *s* ); *// set the second* } *// end method setTime  
  
  
 //validate and set hour* public void setHour( int *h* ){  
 if ( *h* >= 0 && *h* < 24 ) hour = *h*;  
 else  
 throw new IllegalArgumentException( "hour must be 0-23" );  
 } *// end method setHour  
 //validate and set minute* public void setMinute( int *m* ){  
 if ( *m* >= 0 && *m* < 60 )  
 minute = *m*;  
 else  
 throw new IllegalArgumentException( "minute must be 0-59" );  
 } *// end method setMinute  
  
 //validate and set second* public void setSecond( int *s* ) {  
  
 if ( *s* >= 0 && *s* < 60 )  
 second = ( ( *s* >= 0 && *s* < 60 ) ? *s* : 0 );  
 else  
 throw new IllegalArgumentException( "second must be 0-59" );  
 } *// end method setSecond  
  
  
 // Get Methods* public int getHour(){  
 return hour;  
 }  
  
 public int getMinute(){  
 return minute;  
 }  
  
 public int getSecond(){  
  
 return second;  
 }  
 *//Method for incrementing Second* public void incrementSecond(){  
 second += 1;  
 if (second == 60){  
 incrementMinute();  
 second = 0;  
 if (minute == 60){  
 incrementHour();  
 minute = 0;  
 if (hour == 24){  
 hour = 0;  
 }  
 }  
 }  
 }  
 *//Method for incrementing Minute* public void incrementMinute(){  
 minute += 1;  
 }  
 *//Method for incrementing Hour* public void incrementHour(){  
 hour += 1;  
 }  
 *//convert to String in universal-time format (HH:MM:SS)* public String toUniversalString(){  
  
 return String.*format*("%02d:%02d:%02d", getHour(), getMinute(), getSecond() );  
 }  
  
 *//convert to String in standard-time format (H:MM:SS AM or PM)* public String toString(){  
 return String.*format*( "%d:%02d:%02d %s",  
 ( (getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12 ),  
 getMinute(), getSecond(), ( getHour() < 12 ? "AM" : "PM" ) );  
 } *// end method toString*} *// end class Time2*

**Test Program:**

package com.company;  
  
*//Creating the test class Time2Test*public class Time2Test {  
  
 *//Main method* public static void main(String[] *args*) {  
  
 *//Creating object for the validation of next minute incrementation* System.*out*.println("Validating Next Minute Increment");  
 System.*out*.println("--------------------------------");  
 Time2 time1 = new Time2();  
 *//Setting time* time1.setTime(11,34,59);  
 *//Displaying input time* System.*out*.println("Input Universal Time: 11:34:59");  
 System.*out*.println("Input Standard Time: 11:34:59 AM");  
 *//Incrementing second by one* time1.incrementSecond();  
 System.*out*.println("After incrementing,");  
 *//Printing output in universal form* System.*out*.println("Universal Time: " + time1.toUniversalString());  
 *//Printing output in standard form* System.*out*.println("Standard Time: " + time1.toString());  
  
 System.*out*.println();  
  
 *//Creating an object for the validation of next hour incrementation* System.*out*.println("Validating Next Hour Increment");  
 System.*out*.println("--------------------------------");  
 Time2 time2 = new Time2();  
 *//Setting time* time2.setTime(11,59,59);  
 *//Displaying input time* System.*out*.println("Input Universal Time: 11:59:59");  
 System.*out*.println("Input Standard Time: 11:59:59 AM");  
 *//Incrementing second by one* time2.incrementSecond();  
 System.*out*.println("After incrementing,");  
 *//Printing output in universal form* System.*out*.println("Universal Time: " + time2.toUniversalString());  
 *//Printing output in standard from* System.*out*.println("Standard Time: " + time2.toString());  
  
 System.*out*.println();  
  
 *//Creating an object for the validation of next day increment* System.*out*.println("Validating Next Day Increment");  
 System.*out*.println("--------------------------------");  
 Time2 time3 = new Time2();  
 *//Setting time* time3.setTime(23,59,59);  
 *//Displaying input time* System.*out*.println("Input Universal Time: 23:59:59");  
 System.*out*.println("Input Standard Time: 11:59:59 PM");  
 *//Incrementing second by one* time3.incrementSecond();  
 System.*out*.println("After incrementing,");  
 *//Printing output in universal form* System.*out*.println("Universal Time: " + time3.toUniversalString());  
 *//Printing output in standard form* System.*out*.println("Standard Time: " + time3.toString());  
 }  
}

**Output Screenshot:**

****

****

**Deliverables**

Compile a single word file with all codes and their screenshots and upload on the link provided on LMS.

## To Receive Credit

1. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
2. The lab time is not intended as free time for working on your programming/other assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the other assignments.