**Abdul Ghaffar Kalhoro**

**Reg: No: 194699**

**Class: BSCS-6C**

**Object Oriented ProgrammingLab 04: Classes & Objects II**

**Date: March 14th, 2017**

**Activity One: Source code for corrected Method:**

**class Time**

**{**

**private int hour; // 0 - 23**

**private int minute; // 0 - 59**

**private int second; // 0 - 59**

**Time(){**

**setTime(0,0,0);**

**}**

**Time(int h){**

**setTime(h,0,0);**

**}**

**Time(int h, int m){**

**setTime(h,m,0);**

**}**

**Time(int h, int m, int s){**

**setTime(h,m,s);**

**}**

**// 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**

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

**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);**

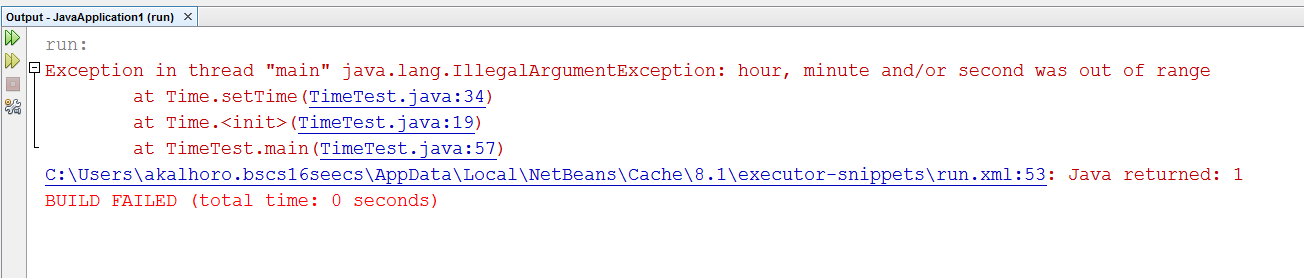
**System.out.println(t1);**

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

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

**}**

**}**

****

**ActivityOne corrected Method source code:**

**class Time**

**{**

**private int hour; // 0 - 23**

**private int minute; // 0 - 59**

**private int second; // 0 - 59**

**Time(){**

**setTime(0,0,0);**

**}**

**Time(int h){**

**setTime(h,0,0);**

**}**

**Time(int h, int m){**

**setTime(h,m,0);**

**}**

**Time(int h, int m, int s){**

**setTime(h,m,s);**

**}**

**// 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**

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

**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, 25); // Set Time to 23:40:25**

**// 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);**

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

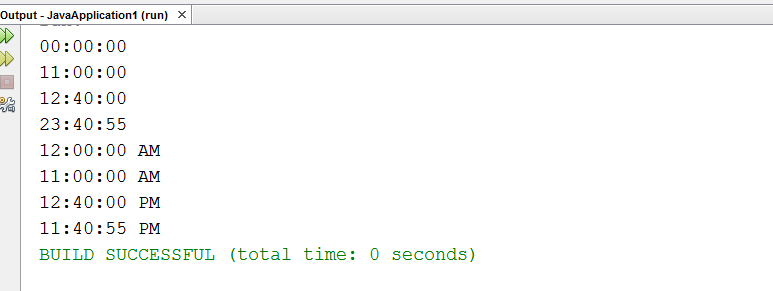
**System.out.println(t2);**

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

**}**

**}**

**OUTPUT**

****

**ActivityTwo:**

// 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);

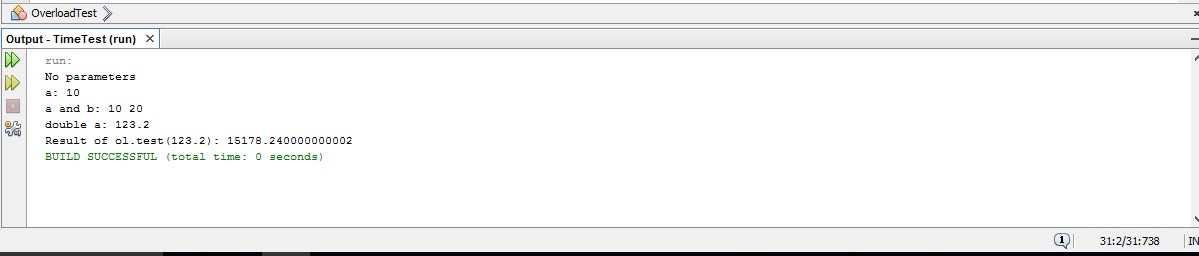
ol.test(10, 20);

result = ol.test(123.2);

System.out.println("Result of ol.test(123.2): " + result);

}

}



**ActivityThree:**

void test(int a) {

System.out.println("a: " + a);

}

// Overload test for one integer parameter.

int test(int a) {

System.out.println("a: " + a);

return 0;

}

**Here repetition of same parameter function is not allowed, therefore the overloading of parametric function is not allowed. As the different return type cannot affect.**

void test(int a, int b) {

System.out.println("a and b: " + a + " " + b);

}

// Overload test for two integer parameters.

void test(int b, int a) {

System.out.println("a and b: " + a + " " + b);

**The number of parameters are repeat again so it is not allowed in java to use same type of repeated parameters so just change in order to make it correct.**

**Task\_1**

**public class PerfectNO {**

**public static void main(String args[]) {**

**int counter;**

**// printing perfect number together with factors.**

**for (int number = 1; number <= 1000; number++) {**

**if (isPerfect(number)) {**

**System.out.printf("\nThe perfect number is %d with factors: ", number);**

**//for loop used for finding and printing of factors.**

**for(counter=1;counter<number;counter++){**

**if(number%counter==0)**

**System.out.printf("%d ",counter);**

**} //end 2nd for loop**

**} //end if statement**

**} //end 1st for loop**

**} // end main method**

**// Method isPerfect for finding perfect number**

**public static Boolean isPerfect(int number) {**

**int sum = 0;**

**//for loop used for perfect number.**

**for (int factor = 1; factor < number; factor++) {**

**if (number % factor == 0) //condition for perfect numbers.**

**{**

**sum += factor; //finding sum of factors.**

**}**

**}**

**if (sum == number) //**

**{**

**return true;**

**} else {**

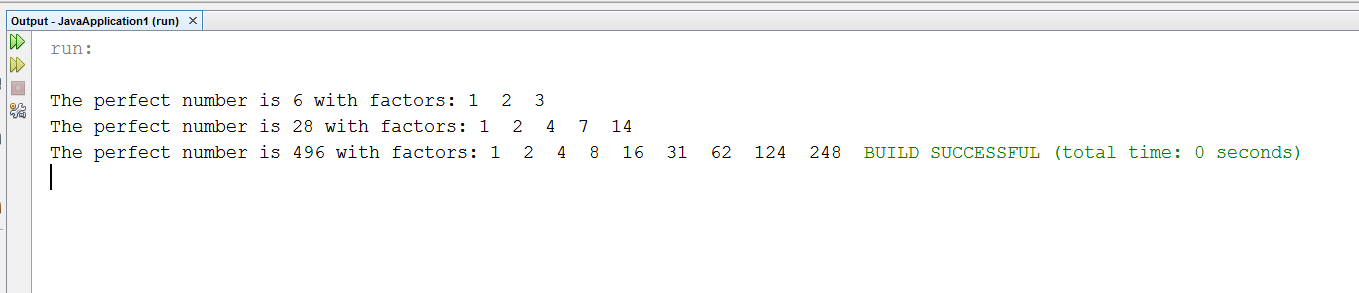
**return false;**

**}**

**} // end method Boolean**

**} // end class**

**OUTPUT**

****

**Task-2**

**//Class of saving account.**

**public class SavingsAccount {**

**static double annualInterestRate = 0.0;**

**private double savingsBalance;**

**//data fields**

**public SavingsAccount() {**

**savingsBalance = 0.0;**

**annualInterestRate = 0.0;**

**}**

**//constructor**

**public SavingsAccount(double initBalance) {**

**savingsBalance = initBalance;**

**}**

**//method with no return type**

**static void modifyInterestRate(double interestRate) {**

**annualInterestRate = interestRate;**

**}**

**//method for counting**

**public void calculateMonthlyInterest() {**

**savingsBalance = savingsBalance + (savingsBalance \* annualInterestRate / 12);**

**}**

**//method that returns the saving balance.**

**public double getBalance() {**

**return savingsBalance;**

**}**

**public static void main(String args[]) {**

**SavingsAccount saver1 = new SavingsAccount(2000);**

**SavingsAccount saver2 = new SavingsAccount(3000);**

**//Calling a method with interesst rate 4%.**

**SavingsAccount.modifyInterestRate(.04);**

**System.out.println("The initial amounts of Saver1 and Sever2 are: ");**

**System.out.println("Saver1 = "+saver1.getBalance());**

**System.out.println("Saver2 = "+saver2.getBalance());**

**saver1.calculateMonthlyInterest();**

**saver2.calculateMonthlyInterest();**

**System.out.printf("\n"+"For interest rate 4%%");**

**System.out.printf("\nAmount of Saver1 is: %.2f ", saver1.getBalance());**

**System.out.printf("\nAmount of Saver2 is: %.2f", saver2.getBalance());**

**//Calling a method with interesst rate 5%.**

**SavingsAccount.modifyInterestRate(.05);**

**saver1.calculateMonthlyInterest();**

**saver2.calculateMonthlyInterest();**

**System.out.printf("\n\n"+"For interest rate 5%%");**

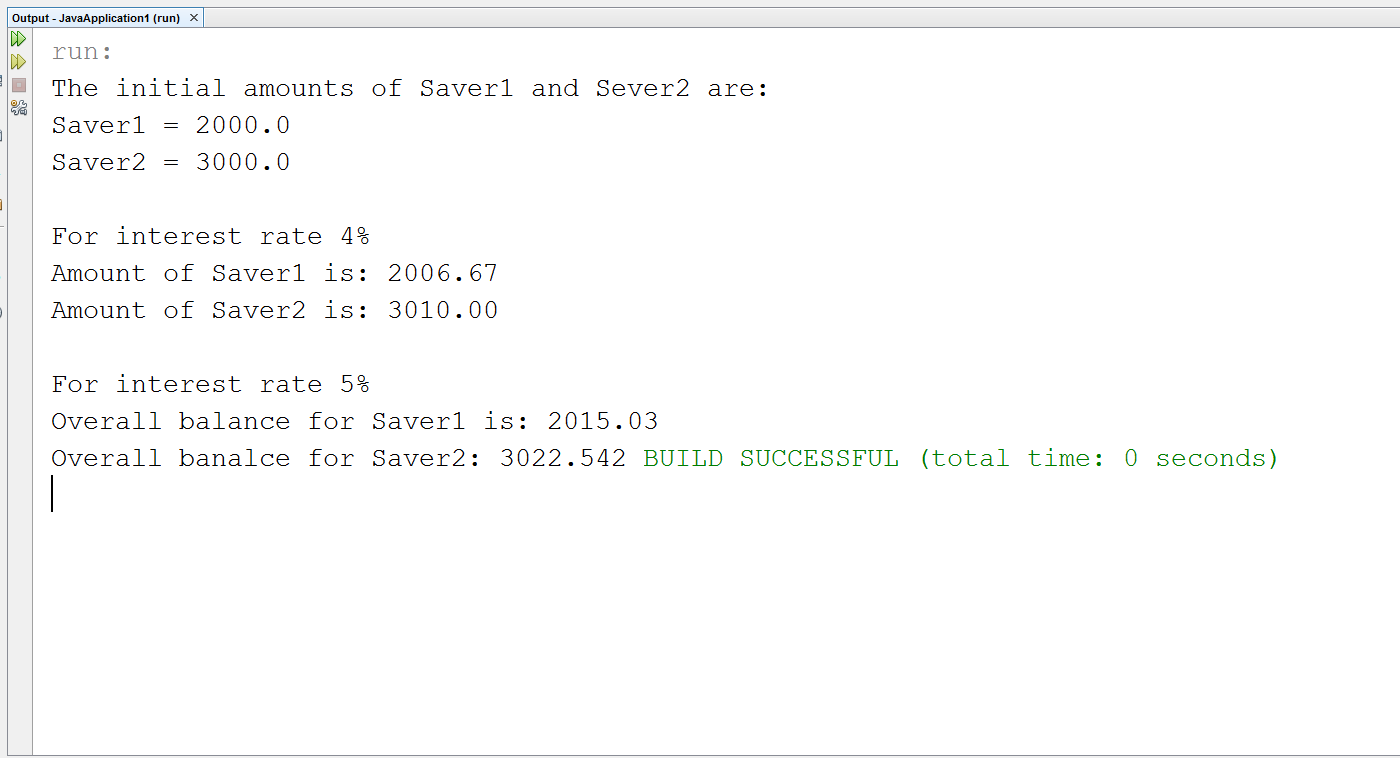
**System.out.printf("\nOverall balance for Saver1 is: %.2f", saver1.getBalance());**

**System.out.printf("\nOverall banalce for Saver2: %.3f ", saver2.getBalance());**

**}**

**}**

**OUTPUT**

****

**Task-3**

FOR CLASS Time2

public class Time2

{

private int hour; // 0 - 2

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 Method

// 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

public void incrementMinute(){

minute++;

if(minute == 60){

incrementHour(); //incrementing to the next hour if minutes become 60

minute = 0; //reseting the minutes

}

}

public void incrementHour(){

hour++;

if(hour == 24){

hour = 0; //reseting the hours

}

}

public void tick(){

second++;

if(second == 60){

incrementMinute(); //incrmenting by one hour

second = 0; //reseting the seconds

}

}

} // end class Time2

**FOR CLASS Test**

public class Test {

public static void main(String[] args){

Time2 object = new Time2(0,0,0);

for(int sec = 0; sec <= 86400; sec++){

object.tick();

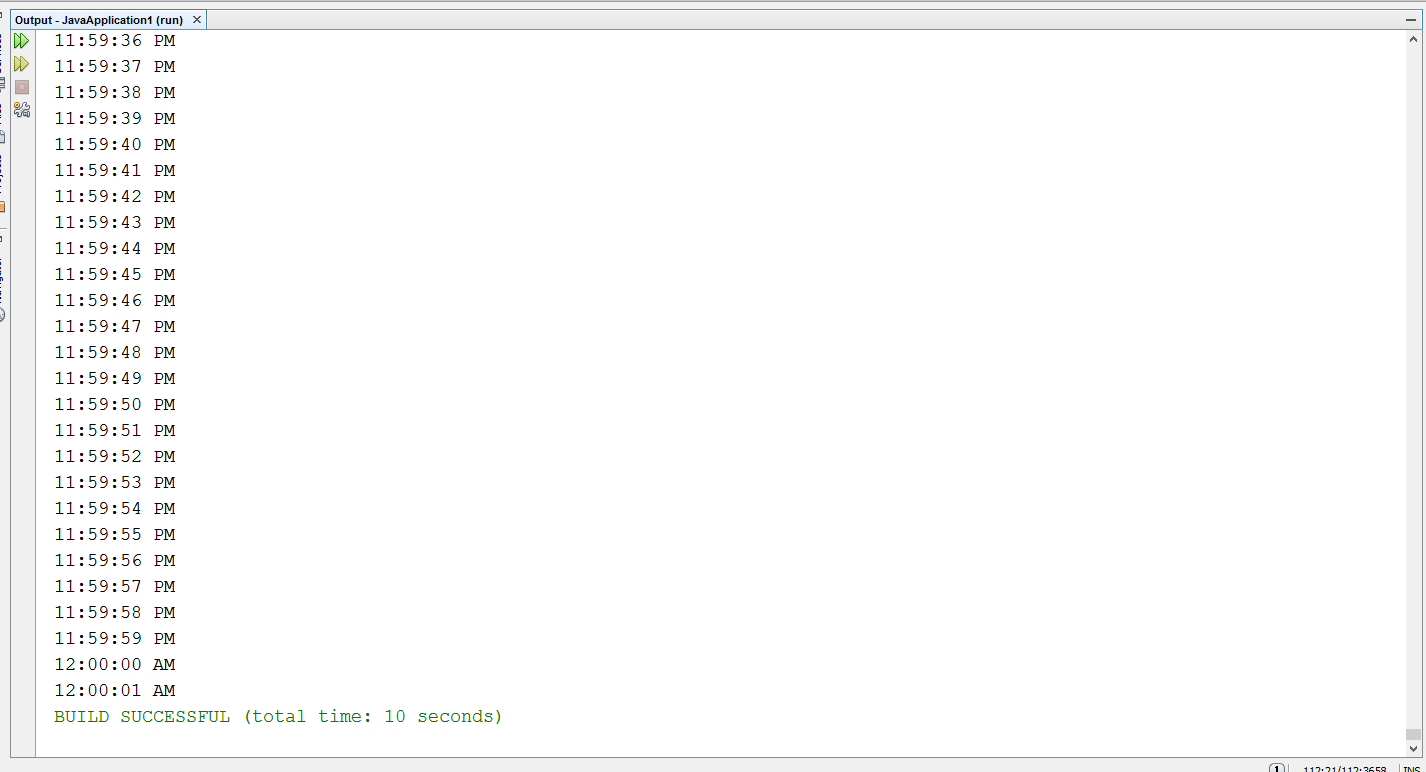
System.out.println(object);

}

}

}

**OUTPUT**

****