**Home Assignment – 6  
Zhenhua (Aaron) Yang**

**Part - I**

**Type your answers to the following questions in a word document.**

1. What is the purpose of a constructor? Is there a limit on the number constructors we can have in a user defined class?

Constructor is used to create an object. A class can have any number of constructors.

1. What is the rule in java with regards to the name of the constructor? Do constructors have return type?

The name of the constructor should be the same as the class name. A constructor doesn’t have any return type.

1. What is the difference between parameters and arguments?

The parameters are the variables that can pass the value in a method, the arguments are the data that are passed.

1. Identify the differences between local variable, instance variable and parameter variable

Local variable is the variable that are defined inside the methods and constructors.

Instance variable is the variable that are defined inside the classes, but outside the methods and constructors.

Parameter variable is the variable that are defined inside the parameter of a method or a constructor

1. Differentiate between instance variable and static variable.

Instance variables are created when an object is created and destroyed when an object destroyed.

Static variables are defined with the key word static. We can only have one copy of the static variable in a class.

1. Explain why should instance variables be declared private ideally?

In this way, the instance variables can be only accessed by the desired method and helps give more control of the class.

1. Explain the difference between accessor and mutator methods?

Accessor methods are used to return the values of variables.

Mutator methods are used to update the values of the variables.

1. What is the difference between arguments and parameter variables.

Arguments are the values or objects that are passed into a method, parameter variables are the variables that receive argument and pass it to the methods.

1. Explain the purpose of toString() and equals( ) method.

toString() method is used to return all the elements of an object, while equals method is used to compare two objects.

1. When you are passing objects to methods, is it a good idea to assign the reference to the object directly to the instance variable? Explain your answer.

No. It would only assign the address of the object.

(Each question is 2 points each = 1.5 \* 10 = 15)

**Part - II**

**Work on the following exercises at the end of chapter 7:**

7.18.2 Reading and Understanding Code

#17: accessor.

#18: mutator.

#27: 1

7.18.3 Fill in the code

#29:

// declare federal tax rate constant; value is 0.07

final double FEDERAL\_TAX\_RATE = 0.07;

#31:

public TelevisionChannel( String n, int num, Boolean c)

{

name = n;

number = num;

cable = c;

}

#36:

Public int countDigits()

{

while ( number != 0 )

{

number /= 10;

count++;

}

return count;

}

#37:

public String isCable( TelevisionChannel t )

{

if ( t.getCable() == true )

return “cable”;

else

return “network”;

}

7.18.4 Identify Errors in Code

#39

Parameter name should not be the same as the instance variable.

#46

The month names should not have quotation marks in enum variable.

(3 points each: 2 \* 2 = 4)

7.18.5 Debugging Area – Using Messages from the Java Compiler and Java JVM

#49

In the constructor Grade, there is a return type char. It requires a return value in the body. But a constructor doesn’t need a return type.

Correct code:

Public Grade( char startLetter )

{

letterGrade = startLetter;

}

#52

The parameter of the constructor is the same as the instance variable. So, every time it runs, the variable is signed by itself, which is 0.

Correct code:

Public Grade( char newNumberGrade )

{

numberGrade = newNumberGrade;

}

(3 points each: 2 \* 2 = 4)

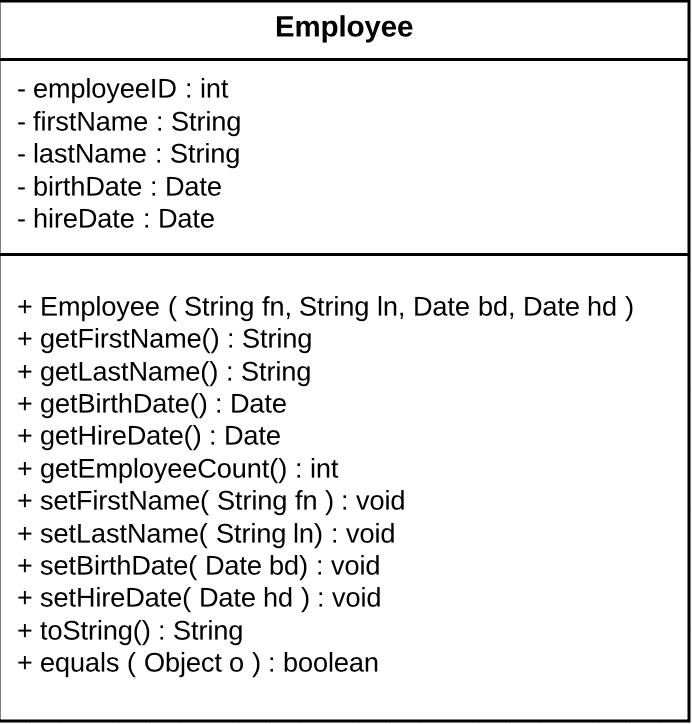
**Part – III: Programming Exercise (60 points)**

/\*\*

\*

\* @author ZhenhuaYang

\*/



public class Employee {

// instance variables

private int employeeID = nextID;

private String firstName;

private String lastName;

private Date birthDate;

private Date hireDate;

static int count;

static int nextID = 1000;

/\*\*

\* constructor

\* @param fn the first name

\* @param ln the last name

\* @param bd the birth date of the employee

\* @param hd the hire date of the employee

\*/

public Employee( String fn, String ln, Date bd, Date hd ){

firstName = fn;

lastName = ln;

birthDate = bd;

hireDate = hd;

count++;

nextID++;

}

// Accessor methods

public String getFirstName(){

return firstName;

}

public String getLastName(){

return lastName;

}

public Date getBirthDate(){

return birthDate;

}

public Date getHireDate(){

return hireDate;

}

public int getEmployeeCount(){

return count;

}

// Mutator methods

public void setFirstName( String fn ){

firstName = fn;

}

public void setLastName( String ln ){

lastName = ln;

}

public void setBirthDate( Date bd ){

birthDate = bd;

}

public void setHireDate( Date hd ){

hireDate = hd;

}

/\*\*

\*

\* @return the data of an Employee object as string

\*/

@Override

public String toString(){

return "\nEmployee Name: " + firstName + " " + lastName

+ "\nEmployee ID: " + employeeID

+ "\nDate of Birth: " + birthDate

+ "\nDate of hiring: " + hireDate;

}

/\*\*

\* compare two Employee objects to see if they are equal to each other.

\* @param o

\* @return

\*/

@Override

public boolean equals( Object o ){

if( !( o instanceof Employee ))

return false;

else{

Employee objEmployee = (Employee) o;

return employeeID == objEmployee.employeeID && firstName == objEmployee.firstName

&& lastName == objEmployee.lastName && birthDate == objEmployee.birthDate

&& hireDate == objEmployee.hireDate;

}

}

}

public class EmployeeTest {

public static void main( String [] args ){

// Create three Employee objects

Employee jake = new Employee( "Jake", "Qi", new Date(2, 23, 1988), new Date(2, 23, 2018));

Employee aaron = new Employee( "Aaron", "Yang", new Date(2, 23, 1986), new Date(1, 23, 2018));

Employee mark = new Employee( "Mark", "Lee", new Date(2, 23, 1988), new Date(4, 12, 2019));

System.out.println( jake.getEmployeeCount()); // test static variable count

// print data of three objects

System.out.println( jake.toString() );

System.out.println( aaron.toString() );

System.out.println( mark.toString() );

// test equals() method

System.out.println( "\nAre Aaron and Mark the same person? " + aaron.equals(mark) );

}

}

import java.util.Calendar;

import java.util.GregorianCalendar;

public class Date {

// Create instant variables

private int day;

private int month;

private int year;

// Date constructor

public Date( int m, int d, int y ){

setYear(y);

setMonth(m);

setDay(d);

}

// Access methods

public int getDay(){

return day;

}

public int getMonth(){

return month;

}

public int getYear(){

return year;

}

/\*\*

\* mutator methods

\* @param d

\*/

// set the day

private void setDay( int d ){

// create GregorianCalendar object

GregorianCalendar cal = (GregorianCalendar) Calendar.getInstance();

switch (month){

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12: {

if( d <=1 || d >= 31 ) // if d<1 or d>31, throw error message.

throw new IllegalArgumentException("Day value must be from 1 to 31 for month " + month);

break;

}

case 4:

case 6:

case 9:

case 11:{

if( d <1 || d > 30 ) // if d<1 or d>30, throw error message.

throw new IllegalArgumentException("Day value must be from 1 to 30 for month " + month);

break;

}

case 2:{

if (cal.isLeapYear(year)){

if( d <1 || d > 29 ) // if d<1 or d>29, throw error message.

throw new IllegalArgumentException("At" + year + ", the day in February must be from 1 to 29");

} else if ( ! ( cal.isLeapYear(year) ) ){

if( d < 1 || d > 28 )

throw new IllegalArgumentException("At" + year + ", the day in February must be from 1 to 28");

}

break;

}

}

day = d; // assign d to day.

}

// set month

private void setMonth( int m ){

if( m < 1 || m > 12 ) // if m<1 or m>12, throw error message.

throw new IllegalArgumentException("Month must be greater than 0 or smaller than 13" );

month = m;

}

// set month

private void setYear( int y ){

if( y < 1582 ) // if y<1582, throw error message.

throw new IllegalArgumentException( "Year must be greater than or equal to 1582");

year = y;

}

// toString method

@Override

public String toString(){

return ( month + "/" + day + "/" + year);

}

// equals method

@Override

public boolean equals( Object o ){

if( ! ( o instanceof Date ))

return false;

else{

Date objDate = ( Date ) o;

return day == objDate.day && month == objDate.month && year == objDate.year;

}

}

}

