***CSC 3020***

***Java Programming***

**Assignment 03**

**50 points**

**Due 03/02/2021 (11:45 A.M.)**

Assignment Objectives:

■■ To demonstrate how to define classes and create objects.

■■ To use UML graphical notation to describe classes and objects.

■■ To distinguish between object reference variables and primitive-datatype variables.

■■ To distinguish between instance and static variables and methods.

■■ To define private data fields with appropriate getter and setter methods.

■■ To create immutable objects from immutable classes to protect the contents of objects.

■■ To use the keyword **this** to refer to the calling object itself.

**Solution to this assignment will not be posted on Canvas; however, any question can be discussed in the class upon request of a student.**

All assignments must be submitted by the Canvas. **No email or hard copy** is accepted. You must follow the following format:

1. For non-programming questions, use a word file to type your answers. Don’t use the text box on the Canvas to answer the questions or to write comments, we will not read it.
2. State your answer clearly.
3. For programming questions, include only the source file for each problem.
4. Submit your file to the Canvas. You must submit your assignment on time; otherwise, you will receive zero. In addition, you cannot submit your file more than one time.
5. There will be several folders on the Canvas. You need to upload your file(s) using the correct folder on the Canvas.
6. Name each file: “Assignment Number(Question number(s))”.
7. To upload your file(s):

* In Course Navigation, click the Assignments link.
* Click the title of the assignment.
* Click the **Submit** Assignment button.
* Add **File**. ...
* Add Another **File**. ...
* **Submit** Assignment. ...
* View **Submission**.

**It is your responsibility to make sure that each file is uploaded correctly. If you uploaded a wrong file, you receive zero; files will not be accepted after due date even if you have a prove that the file is created before the due date.**

**Make sure you review the Cheating & Plagiarism policy on Canvas.**

**Answer questions 1 to 3 on a word file; write a program for each of Q.4 - Q.7.**

**Q01. (10 points - 1 point each)**

1. When will a class have a default constructor?

A class will have a default constructor only if no constructors are explicitly defined in the class.

1. What is an anonymous object?

An anonymous object is an object that is only used once and has no reference.

1. What is NullPointerException?

A runtime error that occurs when invoking a method on a reference variable that has a null value.

1. What is wrong in the following code?

1 **class** Test {

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

3 A a = **new** A();

4 a.print();

5 }

6 }

7

8 **class** A {

9 String s;

10

11 A(String newS) {

12 s = newS;

13 }

14

15 **public** **void** print() {

16 System.out.print(s);

17 }

18 }

On line 3 no string is being placed into the object. Which means line 11 will not be called and it can not print anything.

1. What is the output of the following code?

**public** **class** A {

**boolean** x;

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

A a = **new** A();

System.out.println(a.x);

}

}

The output of the code will be false as the default value of the boolean is false.

1. a. Can you invoke an instance method or use an instance variable from a static method?

No because, static methods cannot invoke or access instance methods or data fields.  
b. Can you invoke a static method or use a static variable from an instance method?   
Yes, static methods and variables can be invoked/accessed from an instance method.

1. In the following code, radius is private in the Circle class, and myCircle is an object of the Circle class. Does the code cause any problems? If so, explain why.

**public** **class** Circle {

**private** **double** radius = 1;

/\*\* Find the area of this circle \*/

**public** **double** getArea() {

**return** radius \* radius \* Math.PI;

}

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

Circle myCircle = **new** Circle();

System.out.println("Radius is " + myCircle.radius);

}

}

No because private variables can still be accessed from within the same class.

1. Describe the difference between passing a parameter of a primitive type and passing a parameter of a reference type. Show the output of the following programs:

**public** **class** Test {

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

Count myCount = **new** Count();

**int** times = 0;

**for** (**int** i = 0; i < 100; i++)

increment(myCount, times);

System.out.println("count is " + myCount.count);

System.out.println("times is " + times);

}

**public** **static** **void** increment(Count c, **int** times) {

c.count++;

times++;

}

}

**class** Count {

**public** **int** count;

**public** Count(**int** c) {

count = c;

}

**public** Count() {

count = 1;

}

}

Passing a parameter of a primitive type involves the passing of the value to the parameter. Passing a parameter of a reference type passes a reference to the value.

Output:

count is 101

times is 0

1. Is the following class immutable?

**public** **class** A {

**private** **int**[] values;

**public** **int**[] getValues() {

**return** values;

}

}

Yes, because all data fields are private their are no mutator methods and their are no accessor methods that return a reference to a mutable object.

1. What is wrong in the following code? Corrected.

1 **public** **class** C {

2 **private** **int** p;

3

4 **public** C() {

5 System.out.println("C's no-arg constructor

invoked");

6 **this**(0);

7 }

8

9 **public** C(**int** p) {

10 p = p;

11 }

12

13 **public** **void** setP(**int** p) {

14 p = p;

15 }

16 }

The statement this(0); on line 6 must be first in the constructor.

**Q02. (4 points)**

Suppose that the class F is defined in (a). Let f be an instance of F.

Which of the statements in (b) are correct?

(a)

**public** **class** F {

**int** i;

**static** String s;

**void** imethod() {

}

**static** **void** smethod() {

}

}

(b)

System.out.println(f.i); **Correct**

System.out.println(f.s); **Correct**

f.imethod(); **Correct**

f.smethod(); **Correct**

System.out.println(F.i);

System.out.println(F.s); **Correct**

F.imethod();

F.smethod(); **Correct**

Instance methods cannot be accessed without a reference variable.

**Q03. (4 points)**

Show the output of the following code:

(a)

**public** **class** Test {

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

**int**[] a = {1, 2};

swap(a[0], a[1]);

System.out.println("a[1] = " + a[1]

+ " a[0] = " + a[0]);

}

**public** **static** **void** swap(**int** n1, **int** n2) {

**int** temp = n1;

n1 = n2;

n2 = temp;

}

}

**Output: a[1] = 2 a[0] = 1**

(b)

**public** **class** Test {

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

**int**[] a = {1, 2};

swap(a);

System.out.println("a[1] = " + a[1]

+ " a[0] = " + a[0]);

}

**public** **static** **void** swap(**int**[] a) {

**int** temp = a[0];

a[0] = a[1];

a[1] = temp;

}

}

**Output: a[1] = 1 a[0] = 2**

(c)

**public** **class** Test {

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

T t = **new** T();

swap(t);

System.out.println("e1 = " + t.e1

+ " e2 = " + t.e2);

}

**public** **static** **void** swap(T t) {

**int** temp = t.e1;

t.e1 = t.e2;

t.e2 = temp;

}

}

**class** T {

**int** e1 = 1;

**int** e2 = 2;

}

**e1 = 2 e2 = 1**

(d)

**public** **class** Test {

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

T t1 = **new** T();

T t2 = **new** T();

System.out.println("t1's i = " +

t1.i + " and j = " + t1.j);

System.out.println("t2's i = " +

t2.i + " and j = " + t2.j);

}

}

**class** T {

**static** **int** i = 1;

**int** j = 1;

T() {

i++;

j = 1;

}

}

**t1’s i = 3 and j = 1**

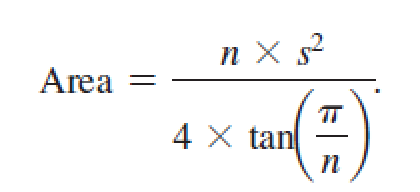
**t2’s i = 3 and j = 1**

**Programming Questions**

**Q04. (8 points)**

In an *n*-sided regular polygon, all sides have the same length and all angles have the same degree. Design a class named **RegularPolygon** that contains:

* A private **int** data field named **n** that defines the number of sides in the polygon with default value **3**.
* A private **double** data field named **side** that stores the length of the side with default value **1**.
* A private **double** data field named **x** that defines the *x*-coordinate of the polygon’s center with default value **0**.
* A private **double** data field named **y** that defines the *y*-coordinate of the polygon’s center with default value **0**.
* A no-arg constructor that creates a regular polygon with default values.
* A constructor that creates a regular polygon with the specified number of sides and length of side, centered at (**0**, **0**).
* A constructor that creates a regular polygon with the specified number of sides, length of side, and *x*- and *y*-coordinates.
* The accessor and mutator methods for all data fields.
* The method **getPerimeter()** that returns the perimeter of the polygon.
* The method **getArea()** that returns the area of the polygon. The formula for computing the area of a regular polygon is



.

Draw the UML diagram for the class then implement the class. Write a test program that creates three **RegularPolygon** objects, created using the no-arg constructor, using **RegularPolygon(6, 4)**, and using **RegularPolygon(10, 4,** **5.6, 7.8)**. For each object, display its perimeter and area.

|  |
| --- |
| **RegularPolygon** |
| n: int  side: double  x: double  y: double |
| RegularPolygon()  RegularPolygon(numSides: int, sideLength: double)  RegularPolygon(numSides: int, sideLength: double, xCord: double, yCord:double)  setN(numSides: int): void  setSide(sideLength: double): void  setX(xCord: double): void  setY(yCord: double): void  getN(): int  getSide(): double  getX(): double  getY(): double  getPerimeter(pObj: RegularPolygon): double  getArea(aObj: RegularPolygon): double |

**Q05. (8 points)**

Design a class named **Time**. The class contains:

* The data fields **hour**, **minute**, and **second** that represent a time.
* A no-arg constructor that creates a **Time** object for the current time. (The values of the data fields will represent the current time.)
* A constructor that constructs a **Time** object with a specified elapsed time since midnight, January 1, 1970, in milliseconds. (The values of the data fields will represent this time.)
* A constructor that constructs a **Time** object with the specified hour, minute, and second.
* Three getter methods for the data fields **hour**, **minute**, and **second**, respectively.
* A method named **setTime(long elapseTime)** that sets a new time for the object using the elapsed time. For example, if the elapsed time is **555550000** milliseconds, the hour is **10**, the minute is **19**, and the second is **10**.

Draw the UML diagram for the class then implement the class. Write a test program that creates three **Time** objects (using **new Time()**, **new** **Time(555550000)**, and **new Time(5, 23, 55)**) and displays their hour, minute, and second in the format hour:minute:second.

(*Hint*: The first two constructors will extract the hour, minute, and second from the elapsed time. For the no-arg constructor, the current time can be obtained using **System.currentTimeMillis().**

The currentTimeMillis method in the System class returns the current time in milliseconds elapsed since the time midnight, January 1, 1970 GMT (Assume the time is in GMT) .

|  |
| --- |
| **Time** |
| hour: long  minute: long  seconds: long |
| Time()  Time(time: long)  Time(h: long, m:long, s:long)  getHour(): long  getMinute(): long  getSeconds(): long  setTime(elapseTime: long): void |

**Q06. (8 points)**

Design a class named **MyInteger**. The class contains:

* An **int** data field named **value** that stores the **int** value represented by this object.
* A constructor that creates a **MyInteger** object for the specified **int** value.
* A getter method that returns the **int** value.
* The methods **isEven()**, **isOdd()**, and **isPrime()** that return **true** if the value in this object is even, odd, or prime, respectively.
* The static methods **isEven(int)**, **isOdd(int)**, and **isPrime(int)** that return **true** if the specified value is even, odd, or prime, respectively.
* The static methods **isEven(MyInteger)**, **isOdd(MyInteger)**, and **isPrime(MyInteger)** that return **true** if the specified value is even, odd, or prime, respectively.
* The methods **equals(int)** and **equals(MyInteger)** that return **true** if the value in this object is equal to the specified value.
* A static method **parseInt(char[])** that converts an array of numeric characters to an **int** value.
* A static method **parseInt(String)** that converts a string into an **int** value.

Draw the UML diagram for the class then implement the class. Write a client

program that tests all methods in the class.

|  |
| --- |
| **MyInteger** |
| value: int |
| MyInteger(v: int):  getValue(): int  isEven(): boolean  isOdd(): boolean  isPrime(): boolean  isEven(v: int): static boolean  isOdd(v: int): static boolean  isPrime(v: int): static boolean  isEven(eObj: MyInteger): static boolean  isOdd(oObj: MyInteger): static boolean  isPrime(pObj: MyInteger): static boolean  equals(v: int): boolean  equals(eqObj: MyInteger): boolean  parseInt(arr: char[]): static int  parseInt(Str: String): static int |

**Q 07. (8 points)**

Design a class named Queue for storing integers. Like a stack, a queue holds elements. In a stack, the elements are retrieved in a last-in first-out fashion. In a queue, the elements are

retrieved in a first-in first-out fashion. The class contains:

* An int[] data field named elements that stores the int values in the queue.
* A data field named size that stores the number of elements in the queue.
* A constructor that creates a Queue object with default capacity 8.
* The method enqueue(int v) that adds v into the queue.
* The method dequeue() that removes and returns the element from the queue.
* The method empty() that returns true if the queue is empty.
* The method getSize() that returns the size of the queue.

Implement the class with the initial array size set to 8. The array size will be doubled once the number of the elements exceeds the size. After an element is removed from the beginning of the array, you need to shift all elements in the array one position the left.

Write a test program that adds 20 numbers from 1 to 20 into the queue then removes these

numbers and displays them.