Java – Conditionals

8.01 Numbers.java

Write a program that reads a floating-point number from the user and then prints out the sign, whether it is *small* or *large*, the number of *digits*, the *whole part* and whether it is odd or even.

static String **getSign**(double n) – returns Positive, Negative or No Sign

static boolean **isSmall**(double n) – returns true if n is within 10 from zero, false otherwise

static boolean **isLarge**(double n) – returns true if n is further than a million from zero, false otherwise

static int **getDigits**(double n) – returns the number of digits (don't include the sign or decimal)

static int **getWhole**(double n) – returns the whole part of the number

static boolean **isOdd**(double n) – returns true if n is odd and false otherwise (be careful -3 % 2 == -1)

public static void **run**(Scanner kb) { // add the prompts and results from the method calls}

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

```
    Scanner kb = new Scanner(System.in);

    // call run(kb) several times to test

}
```

Sample Execution

```
Enter a real number:  3.98
        Positive
        Small as it is within 10 from zero
        Has 3 digits
        The whole part '3' is odd

Enter a real number:  -1000123.4567
        Negative
        Large as it is at least 1,000,000 from zero
        Has 11 digits
        The whole part '-1000123' is odd
```

8.02 ChineseZodiac.java

Write a program that returns the Chinese Zodiac animal given a year (call one static method 5 times passing a Scanner reference). The Chinese Zodiac is based on a twelve-year cycle. The table below provides the animal given year % 12. Don't worry about leap years or switching to the lunar calendar.

The input, output, class and method names must match verbatim.

0	1	2	3	4	5	6	7	8	9	10	11
Monkey	Rooster	Dog	Pig	Rat	Ox	Tiger	Rabbit	Dragon	Snake	Horse	Sheep

static String **chineseZodiac**(int year) – returns the animal based on the year provided

Sample Execution

```
Enter a calendar year: 1913
1913 was the year of the Ox

Enter a calendar year: 2015
2015 was the year of the Sheep

Enter a calendar year: 2022
2022 was the year of the Tiger

Enter a calendar year: 13
13 was the year of the Rooster

Enter a calendar year: 2016
2016 was the year of the Monkey
```

8.03 GPA.java

Write a program that prompts for a letter grade and calculates the numeric grade. Letter grades are A, B, C, D, and F (4 to 0 respectively) and are possibly suffixed with + or -. + increases the value by .3 while a - decreases it by .3. An F does not have a suffix and an A+ does not increase the grade point average.

static double **numericGrade**(String letGrade) – accepts a letter grade as a String and returns the numeric grade. Returns -1 for all invalid grades.

public static void **run**(Scanner kb) { // add the prompt/output and result from numericGrade}

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

```
    Scanner kb = new Scanner(System.in);

    run(kb)

    run(kb)

    run(kb)

    // test the code thoroughly
}
```

Sample Execution

```
Enter letter grade: A+
Numeric grade: 4.0

Enter letter grade: B+
Numeric grade: 3.3

Enter letter grade: C-
Numeric grade: 1.7

Enter letter grade: F
Numeric grade: 0.0

Enter letter grade: E-
Invalid letter grade

Enter letter grade: AA+
Invalid letter grade

Enter letter grade: F-
Invalid letter grade
```

8.04 MilitaryTime.java

Write a MilitaryTime class that represents a time in hours(0 - 23) and minutes(0 – 59).

MilitaryTime() - constructs and initializes a time of 12:00.

MilitaryTime(int h, int m) - initializes a time at h:m. If the hours or minutes are invalid use the default time of noon.

int **compareTo**(MilitaryTime other) – return -1 if the time comes before *other*, 1 if it comes after *other* and 0 if they are the same.

boolean **equals**(Object obj) – return true if the two times are the same and false otherwise. Cast the other object: MilitaryTime other = (MilitaryTime)obj;

String **toString**() – returns the time formatted to hh:mm hours (String.format will be useful).

Sample Data

```
MilitaryTime one = new MilitaryTime();
MilitaryTime two = new MilitaryTime(25, 40);     // invalid goes to noon
System.out.println(one);                         // 1200 hours
System.out.println(two);                         // 1200 hours

System.out.println(one.equals(two));              // should be true
System.out.println(one.compareTo(two) + "\n");    // should be 0

MilitaryTime three = new MilitaryTime(8, 50);
System.out.println(three);                        // 0850 hours
System.out.println(one.equals(three));            // false
System.out.println(one.compareTo(three));         // 1200 is after 0850 so 1
System.out.println(three.compareTo(one));         // 0850 is before 1200 so -1
```

Sample Execution

```
12:00 hours
12:00 hours
true
0

08:50 hours
false
1
-1
```

8.05 Triangle.java

Update the Triangle class by checking the validity of the triangle. Update the mutator methods and constructors by invoking checkTriangleValidity() at the end of the methods. This method will throw an exception if the sides are invalid.

private void **checkTriangleValidity**() -> throw an exception if the sides are invalid

throw new IllegalArgumentException("Invalid Sides");

For a proper triangle, the triangle inequality, as stated in words, literally translates into three inequalities (given that a proper triangle has side lengths a, b, c that are all positive and excludes the degenerate case of zero area):

a + b > c, b + c > a, c + a > b

You may code the expressions above or use a more succinct form. This inequality system can be shown to be

|a - b| < c < a + b

Another way to state it is

max(a, b, c) < a + b + c - max(a, b, c)

implying

2 max(a, b, c) < a + b + c

and thus, the longest side length is less than the semi-perimeter.

Sample Data

```
Triangle test = new Triangle(3, 3, 3);
System.out.printf("%s\n\n", test);

// uncomment to check validity
//test.setSides(5, 8, 3);         // should throw an exception
//test = new Triangle(10, 5, 5); // should throw an exception

// add more test cases
```

8.06 Date.java

Write a Date class that represents a date in month (1 - 12), day (28 – 31) and year.

Date() - constructs and initializes a Date of 01/01/2022.

Date(int month, int day, int year) - initializes a date at month/day/year. If any of the arguments are invalid use the default date. Call daysInMonth to check the upper bounds.

static int **daysInMonth**(int month) returns the number of days(28, 30, 31) in a month(1-12). Try using a switch case. Return -1 for invalid months and don't worry about leap years. Please note this is a class method.

String **getSeason**() – returns the season as winter, spring, summer or fall (all lowercase). Use the following algorithm:

If month is 1, 2, or 3 -> it's winter

Else if month is 4, 5, or 6 -> it's spring

Else if month is 7, 8, or 9 -> it's summer

Else -> it's fall

If month is divisible by 3 and day is greater than 20 -> it's the next season.

String **getDayOfWeek**() – returns the day of the week using Zeller's congruence (Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, or Friday). For the Gregorian calendar, Zeller's congruence is:

$$h = \left(q + \frac{13(m+1)}{5} + k + \frac{k}{4} + \frac{j}{4} + 5j\right) \; mod \; 7$$

where,

h is the day of the week (0=Sat, 1=Sun, 2=Mon, …, 6=Fri)

q is the day of the month

m is the month (3=March, 4=April, 14=February)

k is the year of the century (i.e., year mod 100)

j is the year of the century (i.e., $\frac{year}{100}$)

** it is important to note that January and February are counted as months 13 and 14 of the previous year. If it is January 5th 2022, the algorithm counts the date as 13/05/2021.

** the formula uses integer division so 10/4 results in 2

String **toString**() – returns the date formatted to Weekday mm/dd/yyyy is Season

String.format("%02d", 5) -> returns "05"

Sample Data

```
System.out.println(new Date());
System.out.println(new Date(2, 29, 2022)); // invalid date goes to default
System.out.println(new Date(-1, 5, 500)); // invalid date goes to default
System.out.println();
System.out.println(new Date(3, 20, 2022));
System.out.println(new Date(3, 31, 2050));
System.out.println(new Date(6, 20, 1900));
System.out.println(new Date(6, 21, 1978));
System.out.println(new Date(7, 8, 1980));
System.out.println(new Date(8, 25, 2000));
System.out.println(new Date(10, 15, 1582)); // start of the Gregorian calendar
System.out.println(new Date(11, 7, 2015));
System.out.println(new Date(12, 21, 2030));
```

Sample Execution

```
Saturday 01/01/2022 is Winter
Saturday 01/01/2022 is Winter
Saturday 01/01/2022 is Winter

Sunday 03/20/2022 is Winter
Thursday 03/31/2050 is Spring
Wednesday 06/20/1900 is Spring
Wednesday 06/21/1978 is Summer
Tuesday 07/08/1980 is Summer
Friday 08/25/2000 is Summer
Friday 10/15/1582 is Fall
Saturday 11/07/2015 is Fall
Saturday 12/21/2030 is Winter
```

8.07 RandomWalk.java

Graphically simulate a person randomly walking. A person starts out at the center of a circle and randomly moves up, down, left, or right by one(pixel) with equal probability. The walk ends when the walker reaches the perimeter of the circle (think distance formula). Inside the paintComponent method, if the walker hasn't reached the perimeter, invoke repaint(). Use the starter code from House.java or Picture.java and rename to RandomWalker.java.

Do not call super.paintComponent(g) as this will erase the pixels.

frame.setBackground(Color.BLACK); // if you want to match the color below

Logic help (if needed):

Declare two ints or one Point at the center of your circle (not local variables)

Inside of paintComponent

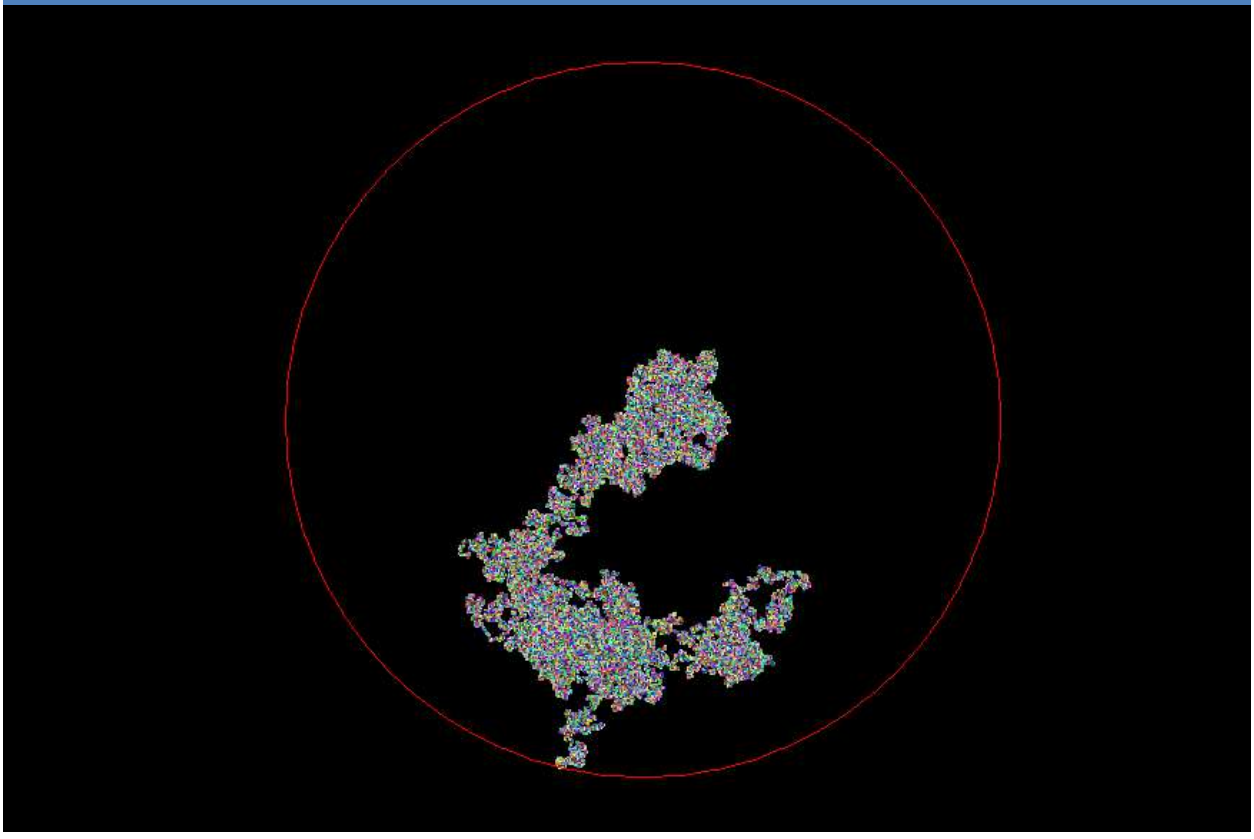
If the distance from the current point to the center < radius

Draw a circle

Move either up/down/left/right w/ equal probability

Invoke repaint()

Sample Execution



8.08 Poker.java (advanced)

Write a program that reads four integers and prints the poker ranking (Four of a Kind, Straight, Three of a Kind, Two Pair, One Pair, High Card). For simplicity, assume the input is an integer in the range of 2 thru 9. You may assume the integers are in increasing order but for a bonus assume any order and don't use sorting.

Sample Execution

```
Enter hand(4 integers): 4 4 4 4
Four of a Kind

Enter hand(4 integers): 7 8 6 9
Straight

Enter hand(4 integers): 2 6 2 2
Three of a Kind

Enter hand(4 integers): 3 5 5 3
Two Pair

Enter hand(4 integers): 2 8 5 2
One Pair

Enter hand(4 integers): 9 5 6 3
High Card
```