**Homework: Review I**

/\*\*

\* Repeatedly asks the user for a positive integer until the user enters

\* one. Returns the positive integer.

\*

\* **@param** in

\* the input stream

\* **@param** out

\* the output stream

\* **@return** a positive integer entered by the user

\*/

private static int getPositiveInteger(SimpleReader in, SimpleWriter out) {

boolean validAnswer = false;

System.***out***.println("Enter a positive integer: ");

String input = in.nextLine();

int n = -69;

// checks to make sure input is valid

while (!validAnswer) {

// checks to make sure value is integer type

if (FormatChecker.*canParseInt*(input)) {

n = Integer.*parseInt*(input);

// checks for positive integer

if (n == 0) {

System.***out***.println("Zero doesn't count as positive");

System.***out***.println("Enter a positive integer: ");

input = in.nextLine();

} else if (n < 0) {

System.***out***.println("I said a POSITIVE integer: ");

input = in.nextLine();

} else {

validAnswer = true;

}

// gives user some sass for not following directions

} else {

System.***out***.println("That's not even a number");

System.***out***.println("Enter a positive INTEGER: ");

input = in.nextLine();

}

}

return n;}

1. a.

|  |  |
| --- | --- |
|  | n = 1  i = 2 |
| While (i < 5) |  |
| n = n + i; | n = 3, 6, 10  i = 2, 3, 4 |
| i = i + 1; | n = 3, 6, 10  i = 3, 4, 5 |
|  | n = 10  i = 5 |

b.

|  |  |
| --- | --- |
|  | i = 2  n = .5 |
| While (i <= 5) |  |
| n = n + 1.0 / i; | i = 2, 3, 4, 5  n = 1.0, 1.33, 1.58, 1.78 |
| i = i + 1; | i = 3, 4, 5, 6  n = 1.0, 1.33, 1.58, 1.78 |
|  | i = 6  n = 1.78 |

c.

|  |  |
| --- | --- |
|  | x = 1.0  y = 1.0 |
| While (x < 1.8) |  |
| y = y / 2.0; | x = 1.0, 1.5, 1.75  y = 0.5, 0.25, 0.125 |
| x = x + y | x = 1.5, 1.75, 1.875  y = 0.5, 0.25, 0.125 |
|  | x = 1.875  y = 0.125 |

d.

|  |  |
| --- | --- |
|  | x = 3  y = 4 |
| While (y > 0) |  |
| x = x + 1; | x = 4, 5, 6, 7  y = 4, 3, 2, 1 |
| y = y – 1; | x = 4, 5, 6, 7  y = 3, 2, 1, 0 |
|  | x = 7  y = 0 |



int total = 0, i = 2;

while (i <= 100) {

if (i % 2 == 0) {

total += i;

}

i++;

}



int total = 0, i = 1;

while (i <= 100) {

total += Math.sqrt(i);

i++;

}



int total = 0, i = 0;

while (i <= 20) {

total += Math.pow(2, i);

i++;

}



int total = 0, i = a;

while (i <= b) {

if (i % 2 == 1) {

total += i;

}

i++;

}



// I can interpret an input however I want. This code will work.

String input = in.nextLine();

int total = 0, i = 0;

while (i < input.length()) {

if (i % 2 == 1) {

total += Integer.parseInt(input.at(input.length() – i));

}

i++;

}



String input = in.nextLine();

int total = 0, i = 1;

while (i < input.length()) {

if (i % 2 == 1) {

total += Integer.parseInt(input.at(i – 1));

}

i++;

}

4.

a. public static int greatestNum(int x, int y) {…}

b. public static int lowestNum(int x, int y, int z) {…}

c. public static boolean isPrime(int x) {…}

d. public static boolean isSubStr(String bigString, String smallString) {…}

e. public static double calcBalance(double initBalance, double interestRate, int years) {…}

f. public static void printBalance(double initBalance, double interestRate, int years) {…}

1. public static void printCalendar(String month, int year) {…}
2. public static void printWeekday(String day, String month, int year) {…}
3. public static int randomInt(int n) {…}

5. Because temp is assigned to the same memory address as A instead of copying the data at that address

6.

public static boolean allTheSame(int x, int y, int z) {

return (x == y && y == z);

}

public static boolean allDifferent(int x, int y, int z) {

return (x != y && y != z && x != z);

}

public static boolean sorted(int x, int y, int z) {

return (x <= y && y <= z);

}

// I hope this works lol I wrote this all in word

public static main(args[]) {

int x = 1, y = 2, z = 3;

if (allTheSame(x, y, z)) {

System.out.println(“All the same”);

}

if(allDifferent(x, y, z)) {

System.out.println(“All different”);

}

if (sorted(x, y, z)) {

System.out.println(“Sorted”);

}