CSE 007

Comprehension (40 points) : 5 mc, 5 quick answer, and 2 snippets of code

Programming : 60 points

Partial credit for everything except mc

Open everything **except Internet and other people at packard 101 4:25-5:40pm**

can use ZyBooks pdf on coursesite

Header comment/variable names/indentation/comments

compiling error - make sure it can at least compile (comment out the error)

A sequence of instructions that solves a problem is called an **algorithm**

The relation between a string's length and string's last index?

The last index is going to determine the length of the string (wrong)

Index of a string starts from 0, so length of string - 1 is going to be the character

example :

```
string example = "daniel"
// here, daniel has 6 characters, but d is 0, and l is 5 (6-1)
```

3. compile time vs runtime error

compile time errors are syntax errors, no compile, runtime errors happen after successful compiling, such as divide by zero error

```
int x;
int y;
x = 2;
y = ((x+1) / 5) + (x / 2);

final value of y = 1, because (x+1)/5 is 0.6, but int will only store 0
```

```
int x;
int y;
x = 6;
y = (1/2) * (x+5);

final value of y = 3; (wrong)
final value of y = 0, because 1/2 is going to be equal to 0. unless i do 1.0 /2 (makes 1.0 a double)
```

```
int x = 77;
int y = 4;
if (x == 77) {
   y = y + 1;
}
if (x < 100) {
   y = y + 1;
}
if (x > 77) {
   y = y + 1;
}
y = y + 1;
```

```
final value of y = 7
correct
```

```
int x = 6;
int y = 2;
if (x < 10) {
   if (y < 5) {
      y = y + 1;
   }
   else {
      y = 7;
   }
}
else {
   y = y + 10;
}
final value of y is 3
```

```
int a = 3, b = 4, c = 5, d = 6;
if( a * b + c != b * c + d ){
  System.out.println(“d: ” + d);
}
System.out.println(“d: ” + 2*d);

d : 6
d : 12
if it was 2+d, then it would print 26, because its adding them together (d is a string)
but if it does (2+d) then it will be d: 8
```

```
char letter1;
char letter2;
letter1 = 'p';
while (letter1 <= 'q') {
   letter2 = 'x';
   while (letter2 <= 'y') {
      System.out.print("" + letter1 + letter2 + " ");
      ++letter2;
   }
   ++letter1;
}

 px py qx qy

remember that lowercase hvae lower value than uppercase
```

```
int num = 10;
while (num <= 15) {
   System.out.print(num + " ");
   if (num == 12) {
    break;
   }
   ++num;
}
System.out.print("Done");

//prints
10 11 12 Done
```

```
String str1 = “2022*”, str2 = “LU”,  str3 = “*”;
int val = 4, val2 =20, x = 5, y = 0;
double val3 = 19.5;
System.out.println(str1 + str2 +  str3 + val2 + (int)val3);

2022*LU*2019



for (int i = 0; i < 3; i++){
```

```
          int newVal = (int) val3;
        }
      System.out.println(“newVal: ” + newVal);

newVal: 19

scope is always a compiler error
you can't declare the int inside the for loop


if( 2.5 < 3.5 ){
        System.out.println("Val: " + val--);
}
System.out.println("Val: " + --val);

Val: 19.5
Val: 17.5




if( x > 4 || x / y < 3){
         System.out.println("test " + ++x);
      }

CE. x/y is division by 0 err

nope, it returns test : 6
```

```
int i;
String str;
str = scnr.nextLine();
for (YYY) {
   System.out.print(str.charAt(i));
}

yyy : i = parse.Characterlength(str), --i
wrong
yyy : int i = str.length - 1; i>=0; i-- (or i-=1)
```

```
for ( YYY ) {
   System.out.print(i + " ");
}
(int i = -10, i<=10, i+=2)
```

```
import jav.util.SCanner;
public class patterns{
public static void main(String[] args){
  Scanner input = new Scanner(System.in);
System.out.println("input a positive, odd integer");
int x  = input.nextInt();

if(x%2 ==0){
  System.out.println("Error: inc input");
} else {
    for(int i = 0; i<x; i++){
      for(int j=0; j<x; j++) {
          System.out.print("0");
          }
          for(int k=0; k < i+1; k++){
            System.out.print("X");
          }
          System.out.println();
}}}}

when x=2, x=3, and x=4
x=2 : Error: incorrect input
x=3 : 000X return 000XX return 000XXX
x=4 : Error: incorrect input
x=5 :
```

```
int k=(new Scanner(System.in)).nextInt();  // do not rewrite
for(int i=1; i < k; i++){
  System.out.println("i :"+i);
}

int k=(new Scanner(System.in)).nextInt();
int i = 1;
while(i < k){
  System.out.println("i :" +i);
  i++
}
correct

if(i <k)
do {
  System.out.println("i :" +i);
  i++;
} while (i < k)
}
do while loop will always print the loop value once, so you should use if statement
when i ran it, i didnt need the if statement
```

```
9.  Write a complete program that will output a right triangle based on random height (triangleHeight) composed of a user specified symbol
Enter a character:   %
Enter triangle height: 5

%
% %
% % %
% % % %
% % % % %
Challenge: Give the user an option to input a value for triangleHeight. Don’t forget to validate the input (just triangleHeight should be a
```

```
Generating Random Numbers
the syntax is
int number = (int(Math.random() *(upperbound - baseNum + 1)) + baseNum
by default, math.random returns random double from [0,1)
```

```
Formatting Code Output
for printf
%010d (d is for digit -- integers, the 10 represnends the padding around the number
  (right aligned by default)
%.2f (f is for floating point num -- doulbe or float, the 2 represent number of dec places)
%s (represents string. %-10s represetns a string that is left aligned with 10space of pad)
%n or \n, then \r, then \t


// leveragfe type casting and arithmetic to shift digit/truncate them (cut digit off)
double x = 47.12345
System.out.println((int) x*100)/100.0);
System.out.println((int)(Math.PI*100000)/100000.0);
```

```java
import java.util.Scanner;
public class Vowel {
   public static void main(String[] arg) {
       Scanner scanner = new Scanner(System.in);
       System.out.println("Enter a character : ");
       char ch = scanner.next().charAt(0);
       boolean isVowel;
       switch (ch) {
           case 'a':
           case 'e':
           case 'i':
           case 'o':
           case 'u':
           case 'A':
           case 'E':
           case 'I':
           case 'O':
           case 'U': isVowel = true;break;
           default: isVowel = false;}
       if (isVowel) {
           System.out.println(ch + " is  a Vowel");
       } else {
           if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')){
               System.out.println(ch + " is a Consonant");
          } else {
               System.out.println("Input is not an alphabet");}
       }
       scanner.close();}}
```

Patterns w/ Nested Loops

Given the number of rows from the user, create the following pattern:

```
Enter the number of rows: 4
 3 6 9 12
  6 9 12
   9 12
    12
    12
   9 12
  6 9 12
 3 6 9 12
```

```java
import java.util.Scanner;
public class Pattern{
   public static void main(String[]args){
       Scanner s = new Scanner(System.in);
       System.out.print("Enter the number of rows: ");
       int rows = s.nextInt();
       for(int i=1; i<=rows; i++){
           for(int j=0; j<i; j++){
               System.out.print(" ");
           }
           for(int k=i; k<=rows; k++){
               System.out.print(3*k+" ");
           }

           System.out.println();
       }
       for(int i=rows; i>0;i--){
           for(int j=i; j>0; j--){
               System.out.print(" ");
           }
           for(int k=i; k<=rows; k++){
               System.out.print(3*k+" ");
           }
           System.out.println();
       }
   }
}
```

Input Validation

Conceptually:

1. Use an infinite loop or a boolean flag variable to continuously prompt the user until valid input is received.
2. Use another boolean variable to check Scanner for variable type
   a. If correct type, read in value
      i. Use another boolean expression to check the range (if req'd).
      ii. If ok: end loop
      iii. If out of range: prompt the user again (but why don't you need to clear the Scanner here?)
   b. If incorrect type, clear the Scanner and prompt the user again.

```
do{
   System.out.println("Enter an integer");
   boolean correct = keyboard.hasNextInt();
   if(correct){
      int val = keyboard.nextInt();
      sum += val;
      flag=false;
   }else{
      System.out.println(" invalid input");
      String junk = keyboard.next();
    }
} while(flag);
```

Input Validation (cont.)

To read in 5 values:

```
int count = 0;
do{
 do{
   System.out.println("Enter an integer");
   boolean correct = keyboard.hasNextInt();
   if(correct){
      int val = keyboard.nextInt();
      sum += val;
          ++count;
      flag=false;
   }else{
      System.out.println(" invalid input");
      String junk = keyboard.next();
    }
 } while(flag);
} while(count<5);
```

Convert to another loop:

```
String str = "Hello, 123!";
for (int i = 0; i < str.length(); i++) {
    char c = str.charAt(i);
    if (Character.isLetterOrDigit(c)) {
        System.out.println(c + " is a letter or digit");
    } else {
        System.out.println(c + " is not a letter or digit");
    }
}
```

Unix Commands

```
mostly just
javac program.java
java program
cd
pwd
clear
ctrl c
```

```
type casting - explicit way to change types of variable
a = (double) b / (double) c
b and c were both initialized as ints, a is a double

a = (double) (b+c/2) this wont have decimals because b+c/2 only has the integer part
```

```
Scope
Member Variables (Class Level Scope)

These variables must be declared inside class (outside any function).
They can be directly accessed anywhere in class.

Variables declared inside a method have method level scope and
can’t be accessed outside the method.
Note : Local variables don’t exist after method’s execution is over.

Loop Variables (Block Scope)
A variable declared inside pair of brackets “{” and “}” in a
 method has scope within the brackets only.

variable cant be initalized then declared again in a if statement again as well.
```

```
precedence
increment decrement ++ --
!= not equal
multilpicative * /
remainder %
additive + -
shift << >> >>>
relational < > <= >=
== !=
and &
or ^ // |
logical and &&
logical or ||
```

```
is whitespace
String str = "The quick brown fox jumps over the lazy dog";
for (int i = 0; i < str.length(); i++) {
    char c = str.charAt(i);
    if (Character.isWhitespace(c)) {
        System.out.println(c + " is a whitespace character");
    } else {
        System.out.println(c + " is not a whitespace character");
    }
}
```