Sarajevo School of Science and Technology
CS/IS 160 Programming and Problem Solving II
April 2015

Test 1 Practice Problem Solutions

Problem 1a.

Answer the following questions:

(a) Each class declaration that begins with keyword _____ must be stored in a file that has exactly the same name as the class and ends with the *.java* file-name extension.

Solution: public.

(b) Return type _____ indicates that a method will perform a task, but will not return any information when it completes its task.

Solution: void.

(c) List four primitive types in Java that store integer-type numbers.

Solution: byte, short, int, and long.

Problem 2a: Simple declarations and statements.

Write a declaration for each of the given instance variables below.

(a) Declare a private integer variable named **ONE** that is initialized to 1.

Solution: `private int ONE = 1;`

(b) Declare a public character variable **lastNameInitial** that is set to your last name initial.

Solution: `public char lastNameInitial = 'M';`

(c) Declare a public float variable **PI** and set it to 3.14.

Solution: `public float PI = 3.14f;`

Implement the following code snippets:

(d) Write a **for** loop that increments **count** from its initial value of 0 to 10 in steps of 2.

Solution:

```
for( int count=0 ; count <= 10; count += 2 ) {
      // can output count
}
```

(e) Write a **while** loop that decrements **count** from its last value of 10 back to 0 in steps of 5.

Solution:

```
int count = 10;
while(count > 0) {
      count -= 5;
      // can output count
}
```

(f) Write an **if** statement that check if **count** variable is equal to zero, in which case, it prints out to the system console "Problem 2 is done".

Solution:

```
if( count == 0 ) {
      System.out.println("Problem 2 is done.");
}
```

Problem 3a: Integers with equal digits.

Write a static method **equalDigits** that returns true if a positive input integer has all identical digits, and false otherwise. For example, integer 1111 has all identical digits, while 1024 does not.

Use the following method prototype, where **number** is the input integer:

```
static boolean equalDigits( int number )
```

Solution:

```
static boolean equalDigits(int number) {

	int rightDigit = number % 10; // get rightmost digit
	number /= 10; // shift number to right

	while(number != 0) {
		int newRightDigit = number % 10;
		if( rightDigit != newRightDigit) {
			return false;
		}
		rightDigit = newRightDigit;
		number /= 10;
	}

	return true;
}
```

Problem 4a: The sum of digits of an integer.

Implement a static method **sumDigits** that returns the sum of input number's digits. For example, if the number is 123, then this methods returns 6 (1+2+3). We will only consider positive input integers and zero.

Use the following method prototype, where **number** is the input integer:

```
static int sumDigits( int number )
```

Solution:

```
static int sumDigits(int number) {
	int sum = 0;

	while( number != 0 ) {
		sum += number % 10;
		number /= 10;
	}

	return sum;
}
```

Problem 5a: Implement Point2D class.

We define a class named **Point2D** which models an (x,y) point in 2D space, where x is its horizontal and y is its vertical coordinate. These coordinates are implemented using

private double variables **x** and **y**.

(a) Implement getter and setter for the private variable **x**.

Solution:

```
// part (a)

public double getX() {
      return x;
}

public void setX(double newX) {
      x = newX;
}
```

(b) Define a public instance method **isOrigin**, which returns true if the point equals to the origin (0,0).

Solution:

```
// part (b)

public boolean isOrigin() {
      return (x == 0 && y == 0);
}
```

(c) Implement a public instance method **negatePoint**, which negates coordinates of the point. For example, negate of point (5,-1) will change its coordinates to (-5,1).

Solution:

```
// part (c)

public void negatePoint() {
      x *= -1;
      y *= -1;
}
```

Problem 1b (10 points).

Answer the following questions:

(a) What is the name of the function where every Java program begins execution?

Solution: main.

(b) Keyword _____ creates an object of the class specified to the right of the keyword.

Solution: new.

(c) List two primitive types in Java that store floating-point numbers.

Solution: float and double.

State whether each of the following statements is true or false. If false, explain why.

(d) An `import` declaration is not required when one class in a package uses another one in the same package.

Solution: True.

(e) Variables declared in the body of a particular method are known as *instance variables* and can be used in all methods of the class.

Solution: False. Variables declared in the body of a particular method are local variables, and can only be used in the body of that method.

Problem 2b: Simple declarations and statements (20 points).

Write a declaration for each of the given *instance variables* below (2 point each).

(a) Declare a private character variable named `someCharacter`.

Solution:
```
private char someCharacter;
```

(b) Declare a public byte variable `tenAsByte` that is initialized to 10.

Solution:
```
public byte tenAsByte = 10;
```

(c) Declare a private string variable **myName** that is initialized to your first name.

Solution:
```
private String myName = "CSIS160";
```

Implement the following code snippets below (7 points each).

(d) Write a **for** loop that decrements integer variable **count** from its initial value of 10 to 0 in steps of 2.

Solution:
```
for( int count = 10; count >= 0 ; count -= 2) ;
```

(e) Write a **while** loop that prints out all even integer numbers from 0 to 100.

Solution:
```
int i = 0;
while( i <= 100 ) {
      if( i % 2 == 0 ) {
            System.out.print(i + " ");
      }
      i++;
}
```

Problem 3b: Rising number (20 points).

A rising number is a positive integer where each digit is greater than or equal to any digit appearing to its left. All of the single digit numbers are rising, as are numbers such as 12567, 144555899, and 888888. To be clear, the number 17689 is not rising, because the second digit 7 is followed by a 6.

Write a static method `risingDigits` that returns true if the input integer is a rising number, and false otherwise.

Use the following method prototype, where `number` is the input integer:
```
static boolean risingDigits( int number )
```

Solution:
```
static boolean risingDigits(int number)
{
    int rightDigit = number % 10; // get the last digit
```

```
    number /= 10;                  // shift digits to right

    while(number > 0) // while there are nonzero digits
    {
        int leftDigit = number % 10;  // get the next digit
        if(leftDigit > rightDigit)    // left greater than right
            return false;

        number /= 10;
        rightDigit = leftDigit; // shift left digit to the right
    }

    return true; // true if all digits are rising
}
```

Problem 4b: Harmonic number (20 points).

The harmonic number of positive integer `n` is defined as the sum of the reciprocals of the first `n` positive integers. For example, the harmonic number of 5 is given by the sum:

$$harmonic(5) = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5}$$

Write a static method `harmonic` that takes an integer number as the input and returns its harmonic number value.

Use the following method prototype, where `number` is the input integer:

```
static double harmonic( int number )
```

Solution:

```
static double harmonic(int number) {

      // initialize harmonic result variable
      double harmonic = 0;

      for(int i=1; i <= number; i++) {
            // here we need to divide 1.0 or cast to double
            harmonic += 1.0/i;
      }

      return harmonic;
}
```

Problem 5b: Rational number class (30 points, each part 6 points).

We define a class named `Rational` that implements the basic methods and operations for rational numbers (numbers that can be represented as a fraction of two integers *a*/*b*, where *a* is the numerator and *b* is the denominator). Note that the denominator cannot be set to zero, since division by zero is not well defined in ordinary arithmetic. Therefore, we need to print an error message to the console in case that the denominator is set to 0.

We have the following Java class definition with empty constructor and methods:

```
// class definition
public class Rational {

     // instance variables
     private int num;
     private int den;

     // constructor (part a)
     public Rational(int num, int den) {}

     // getters and setters (part b and c)
     public int getNum() {}
     public int getDen() {}

     public void setNum(int num) {}
     public void setDen(int den) {}

     // public instance methods (part d and e)
     public boolean isPositive() {}
     public void invert() {}

}
```

(a) Implement the constructor with two input arguments for `num` and `den`.

Solution:

```
// constructor (part a)

public Rational(int num, int den) {

   this.num = num;
   if( den == 0 ) {
      System.out.println("Error: den cannot be zero!");
   }
   else {
      this.den = den;
   }
}
```

(b) Implement the getter method for the private variable `num`.

Solution:

```
public int getNum() {
    return num;
}
```

(c) Implement the setter method for the private variable `num`.

Solution:

```
public void setNum(int inputNum) {
    num = inputNum;
}
```

(d) Implement the public instance method `isPositive`, which returns true if the rational number is positive, and false otherwise.

Solution:

```
public boolean isPositive() {
    return (num/den) > 0;
}
```

(e) Implement the public instance method `invert`, which inverts the rational number. For example, inversion of rational number 5/2 will change it to 2/5.

Solution:

```
public void invert() {

    if( den == 0 ) {
        System.out.println("Error: cannot invert rational number
with zero numerator!");
        return;
    }

    // if numerator is not zero, then swap num and den
    int temp = num;
    num = den;
    den = temp;

}
```