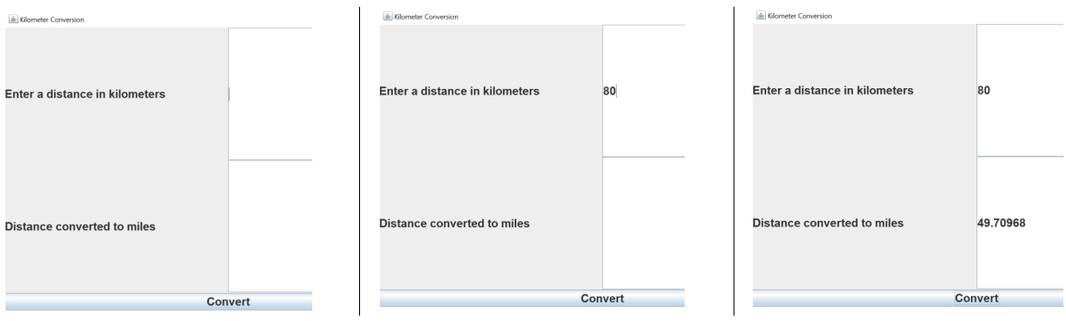
QUEENS COLLEGE Department of Computer Science
CSCI 212 Final Exam Fall 2019 12.17.19

Instructor: Alex Ryba
08.30am – 10.30am, Tuesday, December 17, 2019

1. The Java `class Metric` displays and operates a GUI called Kilometer Conversion that turns kilometer measurements into miles. The class has exactly three methods. You will write these methods.

```
public class Metric extends JFrame implements ActionListener {
  JTextField input, output;
  static Double conversion = 0.621371;    // the number of miles in 1 kilometer
  //  (a)   main method to create the gui --- omitted
  //  (b)   constructor to set up the gui --- omitted
  //  (c)   method to respond to user actions --- omitted
}
```

The following three images show the GUI window as it appears initially, as it appears after the user types a distance in kilometers (here 80), and after the user presses the Convert button.



(a) Write the main method.

Answer:

```
public static void main(String args[]) {
  Metric m = new Metric();
  m.setVisible(true);
}
```

(b) Write the constructor.

Answer:

```
public Metric() {
  super();
  setSize(1200, 800);
  setTitle("Kilometer Conversion");
  input = new JTextField(50);
  output = new JTextField(50);

  Container content = getContentPane();
  content.setLayout(new BorderLayout());

  JPanel p = new JPanel();
  content.add(p, BorderLayout.CENTER);
  p.setLayout(new GridLayout(2, 2));
  p.add(new JLabel("Enter a distance in kilometers"));
  p.add(input);
  p.add(new JLabel("Distance converted to miles   "));
  p.add(output);

  JButton b = new JButton("Convert");
  b.addActionListener(this);
  content.add(b, BorderLayout.SOUTH);
}
```

(c) Write the method that programs the response when the user presses the Convert button.

Answer:

```
public void actionPerformed(ActionEvent e) {
  try {
    Double miles = Double.parseDouble(input.getText()) * conversion;
    output.setText("" + miles);
  } catch (Exception ex) {}
}
```

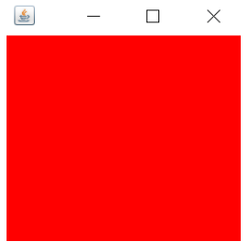
2. The 3 functions in the following class have been omitted. You will write them.

```
// Code for the class Functions

public class Functions {
  public static void main(String args[]) {
    Scanner scan = new Scanner(System.in);
    System.out.println("How old are you? ");
    String input = scan.next();
    int age = getAge(input);                  //  (a)
    if (age == -1) redSquare();               //  (b)
    else {
      Double []arrayD = new Double[age];
      String []arrayS = new String[age];
      for (int i = 0; i < age; i++) {
        Double d = Math.random();
        arrayD[i] = d;
        arrayS[i] = "" + d;
       }
      System.out.println(lastElement(arrayD)); // (c)
      System.out.println(lastElement(arrayS));
    }
  }
  // (a) function   getAge       omitted
  // (b) function   redSquare    omitted
  // (c) function   lastElement  omitted

}
```

Comments about the functions:

(a) **getAge** converts the `String input` to an integer.
It returns -1 if `input` is not an integer, is less than 0 or greater than 99. Otherwise it returns the number shown in `String input`.

(b) **redSquare** creates and displays the following GUI (with a red background color) to signal a user error.



(c) **lastElement** is **a single (generic)** function that returns the last element of an array with an unknown base type.

Answer:

(a) Write the function `getAge`.

Answer:

```
private static int getAge(String input) {
  int x;
  try {
    x = Integer.parseInt(input);
    if (x < 0 || x >= 100) return -1;
  } catch (Exception e) {
    return -1;
  }
  return x;
}
```

(b) Write the function `redSquare`.

Answer:

```
private static void redSquare() {
  JFrame f = new JFrame();
  JPanel p = new JPanel();
  p.setBackground(Color.RED);
  f.getContentPane().add(p);
  f.setSize(300,300);
  f.setVisible(true);
}
```

(c)

Answer:

Write the *generic* function `lastElement`.

```
private static <T> T lastElement(T[] array) {
  return array[array.length - 1];
}
```

3. Consider the following Java program.

```
public static void main(String args[]) {
  ArrayList<Integer> a1, a2, a3;
  a1 = new ArrayList<>();   a2 = a1;   a3 = new ArrayList<>();
  a1.add(3);   a3.add(3);
  System.out.println("" + (a1 == a2) + " " + (a1 == a3));  // line (a)
  System.out.println("" + a1.equals(a2));                  // line (b)
  a2 = a3;
  a2.add(1); a2.add(4); a2.add(5); a2.add(9); a2.add(2); a2.add(6);
  for (Integer x:a2) System.out.print(x); System.out.println(); // line (c)
  for (Integer x:a1) System.out.print(x); System.out.println(); // line (d)
  a3 = new ArrayList<>();   a3.add(9);   a3.add(8);   a3.add(7);
  System.out.println("" + a1.get(0) + a2.get(0) + a3.get(0));   // line (e)
  System.out.println(a1.get(0) + a2.get(0) + a3.get(0));        // line (f)
  Iterator<Integer> x = a3.iterator();
  Iterator<Integer> y = a2.iterator();  Iterator<Integer> z = a2.iterator();
  while (x.hasNext()) {                                        // line (g)
    try {
      System.out.print("" + x.next());
      x.next();
    } catch (Exception e) { System.out.println(" No such thing"); }
  }
  while (x.hasNext()) {                                        // line (h)
    try {
      System.out.print("" + x.next() + "," + x.next() + ",");
    } catch (Exception e) { System.out.println(" No such thing"); }
  }
  while (y.hasNext()) {                                        // line (i)
    try {
      System.out.print("" + y.next() + "," + y.next() + ",");
    } catch (Exception e) { System.out.println(" No such thing"); }
  }
  System.out.println("" + (y == z));  // line (j)
}
```

(a) What is the output from the instruction beginning on line (a)?

Answer:

```
true false
```

(b) What is the output from the instruction beginning on line (b)?

Answer:

```
true
```

(c) What is the output from the instruction beginning on line (c)?

Answer:

```
3141592
```

(d) What is the output from the instruction beginning on line (d)?

Answer:

```
3
```

(e) What is the output from the instruction beginning on line (e)?

Answer:

```
339
```

(f) What is the output from the instruction beginning on line (f)?

Answer:

```
15
```

(g) What is the output from the while loop beginning on line (g)?

Answer:

```
97 No such thing
```

(h) What is the output from the while loop beginning on line (h)?

Answer:

(There is no output in this case.)

(i) What is the output from the while loop beginning on line (i)?

Answer:

```
3,1,4,1,5,9, No such thing
```

(j) What is the output from the instruction beginning on line (j)?

Answer:

```
false
```

4. In this problem, you will write Java code for the `class Fraction` that represents fractions like 5/6 or more generally *num*/*den* (where the variables stand for numerator and denominator). The class should have instance variables and methods as given in the following UML diagram.

Fraction
int num, den
String toString() void multiplyBy(Fraction f) void addOn(Fraction f)

You should also provide a 2 parameter constructor that throws a `RuntimeException` in case its second parameter is 0 (since a 0 denominator is illegal). You **do not** need to reduce fractions to lowest terms. As an example, the following application code should work with your implementation.

```
Fraction x = new Fraction(2, 5);
System.out.println(x.toString());   // prints:  2/5
Fraction y = new Fraction(1, 3);
x.addOn(y);    // changes x to the value of x + y = 2/5 + 1/3 = 11/15
System.out.println(x.toString());   // prints 11/15
```

Answer: (Write your implementation of `class Fraction` below.)

```
public class Fraction {
   private int num, den;
   public Fraction(int n, int d) {
       if (d == 0) throw new RuntimeException();
       num = n; den = d;
   }
   public String toString() {
       if (den > 0) return "" + num + "/" + den;
       else return "" + (-num) + "/" + (-den);
   }
   void multiplyBy(Fraction f) {
       num *= f.num;
       den *= f.den;
   }
   void addOn(Fraction f) {
       num = num * f.den + den * f.num;
       den *= f.den;
   }
}
```

5. In this problem, you will write complete Java code for the `class Ellipse` and its subclass the `class Circle`.

   The `class Ellipse` should use exactly three instance variables `semiMajorAxis`, `focus1` and `focus2` that have types `double`, `Point` and `Point`. You should not implement the `class Point`, but you should use its methods as specified by the following UML diagram.

Point
double x,y
double distanceTo(Point p)

   Your `class Ellipse` must have exactly three methods. The methods are a constructor (with 3 parameters) and methods called `area` and `isOutside`.

   Your `class Circle` should inherit the instance variables and methods of the `class Ellipse`. It should have no extra instance variables and just one extra method: a constructor with 2 parameters that specify its center (a Point) and its radius (a number).

   The following mathematical facts summarize all the information that you need to complete your implementations.

   - As an abbreviation, write a for the value of `semiMajorAxis` and c for half the distance between `focus1` and `focus2`
   - The formula for the area of the ellipse is $\pi a\sqrt{a^2 - c^2}$.
   - A point is outside the ellipse if the sum of its distances to `focus1` and `focus2` is greater than $2a$.
   - A circle with center C and radius r *is an* ellipse with `focus1 = focus2 =` C and `semiMajorAxis =` r.

   (a) Give your implementation for `class Ellipse`.

```
public class Ellipse {
   private double semiMajorAxis;
   private Point focus1, focus2;

   public Ellipse(double a, Point f1, Point f2) {
       semiMajorAxis = a;
       focus1 = f1;
       focus2 = f2;
   }

   public double area() {
      double c = focus1.distanceTo(focus2) / 2;
      return Math.PI * semiMajorAxis * Math.sqrt(
         semiMajorAxis * semiMajorAxis - c * c);
   }

   public boolean isOutside(Point x) {
      return focus1.distanceTo(x) + focus2.distanceTo(x) > 2 * semiMajorAxis;
   }

}
```

   (b) Give your implementation for `class Circle`.

```
public class Circle extends Ellipse {
   public Circle(double r, Point center) {
      super(r, center, center);
   }
}
```