Java Tick 1*

Floating point numbers in Java have a specific binary representation. In this tick you must write a program to unpack the data held in the Java `double` primitive type and display it on the screen. The content presented here is described in greater detail in the Floating Point Computation course. To complete this tick you will probably need to consult external documentation such as the Wikipedia article on IEEE 754[1] for a description of how floating point numbers are stored in memory.

Type in the following program into a file with the correct name and directory structure. Follow the instructions in the comments prefixed with `TODO`. You might find the comments provided before each `TODO` section helpful. When completing this tick please remember to replace `your-crsid` with your username.

```
package uk.ac.cam.your-crsid.tick1star;

public class InspectDouble {

 public static void main(String[] args) throws Exception {

  double d = Double.parseDouble(args[0]);

  // return the bits which represent the floating point number
  long bits = Double.doubleToLongBits(d);

  // Sign bit located in bit 63
  // Suggested Mask 0x8000000000000000L
  // Format 1 => number is negative
  // TODO: fill in the XXXX
  boolean negative = ( XXXX ) != 0;

  // Exponent located in bits 52 - 62
  // Suggested Mask 0x7ff0000000000000L
  // format Sum( 2^n * e(n)) - 1023 (binary number with bias)
  // TODO: fill in the XXXX
  long exponent =  XXXX;

  // Mantissa located in bits 0 - 51
  // Mask left as an exercise for the reader
  // format 1 + Sum(2^-(n+1) * m(n) )
  // TODO: fill in the XXXX
  long mantissabits = XXXX;

  double mantissa = mantissaToDecimal(mantissabits);

  System.out.println((negative ? "-" : "") + mantissa + " x 2^" + exponent);
 }

 private static double mantissaToDecimal(long mantissabits) {
  long one = 0x0010000000000000L;
  return (double)(mantissabits + one) / (double)one;
 }
}
```

[1] http://en.wikipedia.org/wiki/IEEE_754-1985

You can test your code by providing a floating point number to your program as the first argument on the command line. For example, the representation of one hundred in the primitive type `double` can be determined using your program as follows:

```
crsid@machine:~> java uk.ac.cam.your-crsid.tick1star.InspectDouble 100.0
1.5625 x 2^6
crsid@machine:~>
```

Once you are happy with your program you should submit it in a jar file named `crsid-tick1star.jar`. The jar file should have the entry point set to `uk.ac.cam.your-crsid.tick1star.InspectDouble`. If you have done this correctly then you should be able to execute the code inside the jar file directly on the command line as follows:

```
crsid@machine:~> java -jar crsid-tick1star.jar 100.0
1.5625 x 2^6
crsid@machine:~>
```

The jar file should have the following contents:

```
META-INF/
META-INF/MANIFEST.MF
uk/ac/cam/your-crsid/tick1star/InspectDouble.class
uk/ac/cam/your-crsid/tick1star/InspectDouble.java
```