MODULE 4 - SHEET 1

```
public class Bases
 { public static void main(String[] args)
    { System.out.printf("65       is %d%n", 65);
      System.out.printf("0x41     is %d%n", 0x41);
      System.out.printf("0101     is %d%n", 0101);
      System.out.printf("(int)'A' is %d%n", (int)'A');
    }
 }
```

```
public class LongIntro
 { public static void main(String[] args)
    { int jack;
      jack = 2000*2000*2000;
      System.out.printf("jack = %d%n", jack);

      long jill;
      jill = 2000L*2000L*2000L;
      System.out.printf("jill = %d%n", jill);
    }
 }

// This yields:
//
// jack = -589934592
// jill = 8000000000
```

```
public class StringIntro
 { public static void main(String[] args)
    { String a = new String("Jack");
      String b = "Jill";
      String c = a + " and " + b;
      System.out.printf("c is %s%n", c);
      System.out.printf("Length of c is %d%n", c.length());
      if (a.compareTo(b)<0)
         System.out.printf("%s%n", a);
      else
         System.out.printf("%s%n", b);
    }
 }

// This yields:
//
// c is Jack and Jill
// Length of c is 13
// Jack
```

```
public class FormatIntro
 { public static void main(String[] args)
    { System.out.printf("%7d%n", 1234);
      System.out.printf("%s%n", fmtInt(1234,7));
      System.out.printf("%s%n", fmtInt(1234,3));
      System.out.printf("%s%n", fmtInt(1234,-2));
    }

   private static String fmtInt(int n, int d)
    { String s = String.valueOf(n);
      while (s.length() < d)
         s = " " + s;
      return s;
    }
 }

// This yields:
//
//    1234
//    1234
// 1234
// 1234
```

```
public class Primes
 { private static final int SIZE=600, SQRTSIZE=25;

   public static void main(String[] args)
    { boolean[] primes = new boolean[SIZE];
      for (int i=2; i<SIZE; i++)
         primes[i] = true;

      int next = 2;
      while (next < SQRTSIZE)
       { for(int i= next; i<SIZE/next; i++)
            primes[i*next] = false;
         do
  { next++;
  } while (!primes[next]);
       }

      int line=0;
      for (int i=2; i<SIZE; i++)
         if (primes[i])
          { System.out.printf("%5d", i);
            line++;
            if (line%10 == 0)
               System.out.printf("%n");
          }
    }
 }

// This yields:
//
//     2    3    5    7   11   13   17   19   23   29
//    31   37   41   43   47   53   59   61   67   71
//    73   79   83   89   97  101  103  107  109  113
//   127  131  137  139  149  151  157  163  167  173
//   179  181  191  193  197  199  211  223  227  229
//   233  239  241  251  257  263  269  271  277  281
//   283  293  307  311  313  317  331  337  347  349
//   353  359  367  373  379  383  389  397  401  409
//   419  421  431  433  439  443  449  457  461  463
//   467  479  487  491  499  503  509  521  523  541
//   547  557  563  569  571  577  587  589  593  599
//
// Note that 589 is divisible by 19 so there is a bug
// somewhere.  Where is it?  How can it be fixed?
```

```
public class SwapA
 { public static void main(String[] args)
    { int i=2, j=4;
      swap(i,j);
      System.out.printf("i is %d%n", i);
      System.out.printf("j is %d%n", j);
    }

   private static void swap(int a, int b)
    { int t=a;
      a = b;
      b = t;
      return;
    }
 }
```

```
public class SwapB
 { public static void main(String[] args)
    { int[] fred = {2,4};
      swap(fred);
      System.out.printf("fred[0] is %d%n", fred[0]);
      System.out.printf("fred[1] is %d%n", fred[1]);
    }

   private static void swap(int[] a)
    { int t=a[0];
      a[0] = a[1];
      a[1] = t;
      return;
    }
 }
```

```
public class SwitchTest
 { public static void main(String[] args)
    { for (int n=0; n<12; n++)
       { switch(n)
          { case 8:  System.out.printf("Luckiest of all%n");
                     break;
            case 3:  System.out.printf("Very lucky%n");
                     System.out.printf("though not quite up to 8%n");
                     break;
            case 9:  System.out.printf("Long lasting%n");
                     break;
            case 2:  System.out.printf("Easy%n");
                     break;
            case 1:
            case 5:
```

```
            case 6:
            case 7:  System.out.printf("Of no special interest%n");
                     break;
            case 4:  System.out.printf("Very unlucky, meaning death%n");
                     break;
            default: System.out.printf("Not classified%n");
          }
         System.out.printf("%n");
       }
   }
}
```

```
public class SortProg
 { public static void main(String[] args)
    { int[] vec = {5, 3, 2, 4, 6};
      sort(vec);
      for (int i : vec)
         System.out.printf("%d%n", i);
    }

   private static void sort(int[] v)
    { for (int k=1; k<v.length; k++)
       { int i=k;
         while (i>0 && v[i-1]>v[i])
  { int t = v[i-1];
           v[i-1] = v[i];
           v[i] = t;
           i--;
  }
       }
    }
 }

// This yields:
//
// 2
// 3
// 4
// 5
// 6
```

```
public class Easter
 { private static final int YEAR = 2000;

   public static void main(String[] args)
    { int date = easter(YEAR);
      int day=date/10, month=date%10;
      System.out.printf("Easter %d was on %d/%d/%d%n", YEAR, day, month, YEAR);
    }

   private static int easter(int y)
    { int a, b, c, d, e, f, g, h, i, k, l, m, n, p;
      a = y%19;
      b = y/100;
      c = y%100;
      d = b/4;
      e = b%4;
      f = (b+8)/25;
      g = (b-f+1)/3;
      h = (19*a+b-d-g+15)%30;
      i = c/4;
      k = c%4;
      l = (32+2*e+2*i-h-k)%7;
      m = (a+11*h+22*l)/451;
      n = (h+l-7*m+114)/31;
      p = (h+l-7*m+114)%31;
      return 10*(p+1)+n;
    }
 }

// This yields:
//
// Easter 2000 was on 23/4/2000
```

```
public class FandBsum
 { private static final float PI=3.141593f;

   public static void main(String[] args)
    { int n=0;
      float last=0.0f, next=1.0f/PI;

      while (next > last)
       { n++;
         last = next;
         next += 1.0/(1000*n+PI);
       }
      System.out.printf("Forward sum is %f (%d terms)%n", next, n);

      float sum=0.0f;
      for (int i=n; i>=0; i--)
```

```
        sum += 1.0/(1000*i+PI);
      System.out.printf("Backward sum is %f%n", sum);
    }
 }

// This yields:
//
// Forward sum is 0.330266 (67109 terms)
// Backward sum is 0.329996
```

```
public class RoundingErrors
 { private static final int LIMIT = 16;

   public static void main(String[] args)
    { System.out.printf("    Right    Wrong%n%n");

      int numerator = 2;

      for (int n=1; n<=LIMIT; n++, numerator *= 2)
       { float right = numerator/100.0f;
         System.out.printf("%9.2f", right);
         float wrong = 0.0f;
         for (int i=1; i<=numerator; i++)
    wrong += (float)1/(float)100;
         System.out.printf("%9.2f%n", wrong);
       }
    }
 }

// This yields:
//
//     right    wrong
//
//      0.02     0.02
//      0.04     0.04
//      0.08     0.08
//      0.16     0.16
//      0.32     0.32
//      0.64     0.64
//      1.28     1.28
//      2.56     2.56
//      5.12     5.12
//     10.24    10.24
//     20.48    20.48
//     40.96    40.96
//     81.92    81.92
//    163.84   163.83
//    327.68   327.69
//    655.36   655.69
```

```
public class RootFive
 { private static final int N = 5;

   public static void main(String[] args)
    { double x = (double)N, oldx;
      do
       { oldx = x;
         x = 0.5d*(x+(double)N/x);
```

```
       } while (Math.abs(x-oldx)>1.0e-10d);
      System.out.printf("Root %d is %.14f%n", N, x);
    }
 }

// This yields:
//
// Root 5 is 2.23606797749979
```