1. /** = the number rhino's in r's ancestral tree whose
        father is not known. Precondition: r may be null */
public static int numberNoFather(Rhino r) {
   if (r == **null**) **return** 0;
   int n= numberNoFather(r.mother);
   if (r.father == **null**) **return** 1 + n;
   return numberNoFather(r.father) + n;
}

2(c).

2(a). JFrame: Border-
Layout; JPanel: Flow-
Layout; Box: BoxLay-
out. JFrame: in 0..5;
JPanel: ≥ 0; Box: ≥ 0.
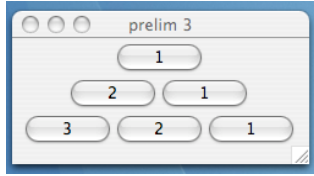
2(b). 1 vertical box and 4 horizontal boxes, so: 5.

3. /** See prelim 3 for the spec */
public static String extract(String s, **int**[] b) {
   String result= "[";
   // invariant: result =   "[" catenated with chars of s whose
   //   indices are given by b[0..k–1], separated by commas
   for (**int** k= 0; k < b.length; k= k+1) {
      // Process b[k]
      if (k != 0)
          result= result + ",";
      result= result + s.charAt(b[k]);
   }
   // result =   "[" catenated with chars of s whose indices
   //   are given by b[0..b.length()–1], separated by commas
   return result + "]";
}

4 (a). import java.util.*;
/** An instance is a superellipse */
public class SuperEllipse {
   // The 3 fields describe a superellipse
   // |x/a|^n + |y/b|^n = 1
   private double a, b, n;

   /** Constructor: an instance of a superellipse
          |x/a|^n + |y/b|^n = 1
       Precondition: a > 0, b > 0, n > 0 */
   public SuperEllipse(**double** a, **double** b,
                         double n)
      { **this**.a= a; **this**.b= b; **this**.n= n; }

   public double getA() { **return** a; }

   public double getB() { **return** b; }

   public double getN() { **return** n; }

   /** = "Superellipse: a = <a>, b = <b>, n = <n>"*/
   public String toString() {
      return "Superellipse: a = " + a + ", b = " + b +
             ", n = " + n;
}   }

public class Ellipse **extends** SuperEllipse {
   /** Constructor: an instance of an ellipse
          (x/a)^2 + (y/b)^2 = 1

       Precondition: a > 0, b > 0 */
   public Ellipse(**double** a, **double** b)
      { **super**(a, b, 2.0); }

   /** = the area of this ellipse = Math.PI * a * b */
   public double area()
      { **return** Math.PI * getA() * getB(); }

   /** = "Ellipse: a = <a>, b = <b>, area = <area>" */
   public String toString() {
      return "Ellipse: a = " + getA() + ", b = " +
             getB() + ", area = " + area();
   }
}

public class Circle **extends** Ellipse {
   /** Constructor: an instance of an ellipse
          (x/r)^2 + (y/r)^2 = 1
       Precondition: r > 0 */

   public Circle(**double** r) { super(r, r); }

   public double getRadius() { **return** getA(); }

   /** = "Circle: radius = <radius>, area = <area>" */
   public String toString() {
      return "Circle: radius = " + getRadius() +
             ", area = " + area();
   }
}

4(b). The first statement of a constructor of a subclass
must be either a call on another constructor in the sub-
class or on a constructor of the superclass. The princi-
ple is that inherited fields should be initialized before
newly declared fields.

(c). A parameter is declared in he header of a method.
A local variable is declared in a method body. A field
or instance variable is declared in a class (without
modifier **static**). A static variable is declared in a class
(with modifier **static**).

4(d). Local variables are created when the frame for
the call is created, before execution of the body.

5. /** return an integer k that satisfies
        b[p..k] <= x < b[k+1..q-1].
     Precondition: b[p..q-1] is sorted */
public static int bsearch(**int**[] b,
                          int p, **int** q, **int** x) {
   int k= p-1; **int** j= q;
   // inv: b[p..k] <= x and b[j..q-1] > x
   while (k+1 < j) {
      int e= (k+j) / 2;
      if (b[e] <= x) k= e;
      else j= e;
   }
   return k;
}