CSE 142, Summer 2008
Final Exam Key

1. Expressions

Expression	Value
`1 + 9 / 2 * 2.0`	`9.0`
`5.0 / (3125 % 2) + 2 * (5 / 3)`	`7.0`
`6 % 17 + 9 % 3 + 22 / 4 / 2.0`	`8.5`
`"[" + 2 + 4 * 2.0 + "]" + 3-1`	`"[28.0]3"`
`!(3 < 2) && (4.3 > 3 || 3 < 2)`	`true`

2. Array Mystery

Expression	Final Contents of Array
`String[] a1 = {"a", "b", "c"};` `arrayMystery(a1);`	`["ac", "bb", "cac"]`
`String[] a2 = {"a", "bb", "c", "dd"};` `arrayMystery(a2);`	`["add", "bbc", "cbbc", "ddadd"]`
`String[] a3 = {"z", "y", "142", "w", "xx"};` `arrayMystery(a3);`	`["zxx", "yw", "142142", "wyw", "xxzxx"]`

3. Inheritance Mystery

```
cat 2   dog 1
cat 2
cat

cat 1
cat 2
cat

lion 2   cat 2   dog 1
lion 2   cat 2
lion (or dog)
```

4. **File Processing (three solutions shown)**

```
public static int blackjack(Scanner input) {
   int total = 0;
   while (input.hasNext()) {
      if (input.hasNextInt()) {
         total += input.nextInt();
      } else {
         String token = input.next().toLowerCase();
         if (token.equals("j") || token.equals("q") || token.equals("k")) {
            total += 10;  // jack/queen/king
         } else {
            total += 11;  // ace
         }
      }
      input.next();  // suit
   }

   return total;
}

public static int blackjack(Scanner input) {
   int total = 0;
   while (input.hasNext()) {
      if (input.hasNextInt()) {
         total += input.nextInt();
      } else {
         String token = input.next().toLowerCase();
         if (token.equals("a")) {
            total += 11;
         } else if (token.equals("j") || token.equals("q") || token.equals("k")) {
            total += 10;  // jack/queen/king
         }
      }
   }
   return total;
}

public static int blackjack(Scanner input) {
   int total = 0;
   while (input.hasNext()) {
      if (input.hasNextInt()) {
         total += input.nextInt();
      } else {
         String token = input.next().toLowerCase();
         if (token.equals("a")) {
            total += 11;
         } else {
            total += 10;  // jack/queen/king
         }
      }
      input.next();  // suit
   }

   return total;
}
```

5. **Array Programming (three solutions shown)**

```
public static boolean allPlural(String[] a) {
   for (int i = 0; i < a.length; i++) {
      if (a[i].length() == 0) {
         return false;
      }
      char c = a[i].charAt(a[i].length() - 1);
      if (c != 's' && c != 'S') {
         return false;
      }
   }

   return true;
}

public static boolean allPlural(String[] a) {
   int count = 0;
   for (int i = 0; i < a.length; i++) {
      if (a[i].endsWith("s") || a[i].endsWith("S")) {
         count++;
      }
   }

   if (count == a.length) {
      return true;
   } else {
      return false;
   }
}

public static boolean allPlural(String[] a) {
   for (int i = 0; i < a.length; i++) {
      if (!a[i].toLowerCase().endsWith("s")) {
         return false;
      }
   }
   return true;
}
```

6. Critters (two solutions shown)

```
public class Minnow extends Critter {
    private int cycleLength;
    private int cycleStep;

    public Minnow() {
        cycleLength = 1;
        cycleStep = 0;
    }

    public boolean eat() {
        cycleLength++;
        cycleStep = 0;

        return false;
    }

    public Direction getMove() {
        if(cycleStep == 0) {
            cycleStep++;
            return Direction.SOUTH;
        } else if(cycleStep < cycleLength) {
            cycleStep++;
        } else {
            cycleStep = 0;
        }

        if(cycleLength % 2 == 1) {
            return Direction.EAST;
        } else {
            return Direction.WEST;
        }
    }
}

public class Minnow extends Critter {
    private Direction currHoriz;
    private int cycleLength;
    private int cycleStep;

    public Minnow() {
     currHoriz = Direction.EAST;
     cycleLength = 1;
     cycleStep = 0;
    }

    public boolean eat() {
     cycleLength++;
     cycleStep = 0;

     if(currHoriz == Direction.EAST) {
         currHoriz = Direction.WEST;
     } else {
         currHoriz = Direction.EAST;
     }
     return false;
    }

    public Direction getMove() {
     if(cycleStep == 0) {
         cycleStep++;
```

```
            return Direction.SOUTH;
        } else if(cycleStep < cycleLength) {
            cycleStep++;
            return currHoriz;
        } else {
            cycleStep = 0;
            return currHoriz;
        }
      }
   }
```

7. **Array Programming (six solutions shown)**

```
public static void reverseChunks(int[] a, int size) {
   for (int i = 0; i + size - 1 < a.length; i += size) {
      int left = i;
      int right = i + size - 1;
      while (left < right) {
         int temp = a[left];
         a[left] = a[right];
         a[right] = temp;
         left++;
         right--;
      }
   }
}

public static void reverseChunks(int[] a, int size) {
   for (int i = 0; i < a.length; i++) {
      if (i % size == 0 && i <= a.length - size) {
         for (int j = 0; j < size / 2; j++) {
            int temp = a[i + j];
            a[i + j] = a[i + size - j - 1];
            a[i + size - j - 1] = temp;
         }
      }
   }
}

public static void reverseChunks(int[] a, int size) {
   if (size <= a.length) {
      for (int i = 0; i < a.length; i++) {
         int j = (i - i % size) + size - 1 - i % size;
         if (j > i && j < a.length) {
            int temp = a[i];
            a[i] = a[j];
            a[j] = temp;
         }
      }
   }
}

public static void reverseChunks(int[] a, int s) {
   for (int i = 0; i < a.length / s; i++) {
      for (int j = 0; j < s / 2; j++) {
         int temp = a[i * s + j];
         a[i * s + j] = a[(i + 1) * s - 1 - j];
         a[(i + 1) * s - 1 - j] = temp;
      }
   }
}
```

```
public static void reverseChunks(int[] a, int s) {
   for (int i = 0; i < a.length / s; i++) {
      int[] b = new int[s];
      for (int j = 0; j < s; j++) {
         b[s - 1 - j] = a[i * s + j];
      }

      for (int j = 0; j < s; j++) {
         a[i * s + j] = b[j];
      }
   }
}

public static void reverseChunks(int[] a, int size) {
   for (int i = 0; i <= a.length - size; i += size) {
      for (int j = 0; j < size / 2; j++) {
         int temp = a[i + j];
         a[i + j] = a[i + size - j - 1];
         a[i + size - j - 1] = temp;
      }
   }
}
```