Objectives: MAPS/Dictionaries, Constructors, accessors

Up next: MP5 due : Monday, 8PM

1. **Let's build a game...**

```
public class Sprite {
    private int x,y,dir,shape;
    public void setX(int newX) { if(newX>0) this.x = newX; }
    public int getX() { return x; }
    public int getShape() {return shape;}
    public void move() { if(dir==1) x++; …}

    public void draw() {
      if(shape==1) Zen.drawImage(“InkyGhost.png”, x, y);
      if(shape==2) Zen.drawImage(“Pacman.png”, x, y);
      if(shape==3) Zen.drawImage(“BlinkingDot.png”, x, y);
       //    ... there has to be a better way…
    }
```

```
// Create a constructor to initialize the sprite using a string
whose format is shape,xvalue,yvalue (i.e. comma-separated
values)
```

2. **Write code to create sprites based on the position data in a file “gameData.txt”.**

```
gameData.txt:
      2, 17, 25
      3, 14, 19
      3, 5, 2
      1, 18, 4
      .
      .
      .
```

```
public class Game {
   public static void main(String[] ) {
```

```
    }
}
```

3. Taking our key from biology - **Genetic Algorithms**:

6. **MAPS** (aka **dictionaries**): '*collection*' of associations between key-value pairs.

Examples: dictionaries, phonebooks, color tables, …

7. **Implement Caller ID:** MAPS (Store and retrieve a value for a particular key)

```
public class CallerIdPair {
    public int _____________; // the extension (a unique key)
    public  String  _____________; // the value (can be anything)
}

public class CallerIdMap {

// use an array of pairs
  private


  public ______ add(int phoneNumber, String name) {
    // for now, assume that the extension (the key)
    // has not already been added to this map.
    // better implementations would prevent or
    // remove/replace an existing match.









   }
continued next time …  —->
```

4. **StringBuilder vs. String**

Advantages:

Disadvantages:

5. **StringBuilder vs. String: appending** - the race is on…

Write a procedural program (remember those?) to compare the speed of appending a character to a String (s = s + "!") to a StringBuilder object (s.append("!")).

`System.currentTimeMillis()` might be handy!
From Java Documentation:
`public static long currentTimeMillis()`

Returns the current time in milliseconds represented as the difference (measured in milliseconds) between the current time and midnight, January 1, 1970 UTC.

```
}
```

CS 125 - Lecture 26

Objectives: Constructors, accessors, lists and MAPS

Up next: MP5 due : Monday 8PM

1. Complete the US State class below so that we can create states in the following way :

```
State s1 = new State();

// 52% are democ. voters, 45% are repub. votes. 3% other-
State ptr = new State("Michigan",0.52, 0.45);

State copyOfMichigan = new State(ptr);
```

```
class State {
   private String name
   private double dem; // likelihood of democratic result 0..1
   private double repub; // likelihood of republican result 0..1
   private double other; //likelihood of independent results 0..1
   double getDem()    {                              }
   double getRepub() {                               }
   double getOther() {                               }
   String getName()  {                               }
   String toString() { return getName() + ": " + getDem()
                    + ", " + getRepub() + ", " + getOther(); }
```

6. **Implement Caller ID:** MAPS (Store and retrieve a value for a particular key)

```
   public String get(int phoneNumber) {
      // return "?" if we do not know this extension's name
```

```
   }

public class Driver {
   public static void main (String[] args) {
   }
}
```

2. **Make a list of U.S. States:**

```
   StateList list = new StateList();
   State ptr = new State("Michigan",0.52, 0.45);
   list.add(ptr);
.
.
.

public class StateList {
     private State[] array = new State[0]; // empty array of pointers.
          // Note Each time add is called we'll make a larger array.

     public State getState(int i) { return array[i];}
     public int getSize() { return array.length; }

     public void add(State s) {
        State[] temp = new State[ this.array.length + 1];
       for (int i=0;i<state.length;i++) temp[i] =  array[i];   ???

       temp[ temp.length - 1 ] =s;                              ???
       this.array = temp; // array pointer now looks at new array
     }

     public void addAll(StateList other) {//Spot the error :-)  ???
       for(int x=0; x < other.length;x++)
            add(other.getState(x));
     }
     // returns states where state.repub > 0.5
     public StateList getRepublicanStates() {
          StateList result = new StateList();
          for(int x=0; x< array.length; x++) {
               State state = getState(x);
               if(state.getRepub() > 0.5)
                    result.add( state );
          }
          return result;
     }
     // ---- CONSTRUCTORS ----
     public StateList() { // do nothin'
     }
     public StateList( StateList other) {
          array = new State[ other.getSize() ];
          for(int x=0; x< array.length; x++) {
               array[x] = other.getState(x); // SHALLOW COPY or

               array[x] = _____// DEEP
           }
     }
}
```

```
public class Sprite {
    private int x,y,dir,shape;
    public void setX(int newX) { if(newX>0) this.x = newX; }
    public int getX() { return x; }
    public int getShape() {return shape;}
    public void move() { if(dir==1) x++; …}

    public void draw() {
      if(shape==1) Zen.drawImage("InkyGhost.png", x, y);
      if(shape==2) Zen.drawImage("Pacman.png", x, y);
      //    ... there has to be a better way...
   }

// Create a constructor to initialize the sprite using a string
whose format is xvalue,yvalue (i.e. comma-separated values)




}
```

6. **Write code to create sprites based on the position data in a file.**

```
public class Game {
   public static void main(String[] ) {




   }
}
```

1. **StringBuilder vs. String**

Advantages:

Disadvantages:

2. **StringBuilder vs. String: appending** - the debate is on!

Write a procedural program (remember those?) to compare the speed of appending a character to a String (s = s + "!") to a StringBuilder object (s.append("!")).

`System.currentTimeMillis()` might be handy!

From Java Documentation:

```
public static long currentTimeMillis()
```

Returns the current time in milliseconds represented as the difference (measured in milliseconds) between the current time and midnight, January 1, 1970 UTC.

6. **Let's build a game...**

```
new Ghost(); // creates ghost at (1, random Y position)
new Ghost( new int[] {15,20} );  //ghost at (15,20)

public class Sprite {
    private int x,y,dir,shape;
    public void setX(int newX) { if(newX>0) this.x = newX; }
    public int getX() { return x; }
    public int getShape() {return shape;}
    public void move() { if(dir==1) x++; …}

    public void draw() {
      if(shape==1) Zen.drawImage("InkyGhost.png", x, y);
      if(shape==2) Zen.drawImage("Pacman.png", x, y);
      //    ... there has to be a better way...
   }

// Create a constructor to initialize the sprite using a string
whose format is xvalue,yvalue (i.e. comma-separated values)
```

7. **Complete** `.equals` **and write the two Ghost constructors so we can make ghosts such as :**

```
public class Ghost {
   private static int count=0;
   private static int nextId() {
      count ++; // first ghost will have an id of 1
      return count;
   }
   // each ghost has an x,y and unique id
   private int x=1,y=2+(int)(Math.random()*10);
   private int id = nextId(); // constructor sets id to
                              // a unique value ..

   public String toString() {
      return "Ghost #"+id+" : "+x+","+y;
   }

   public boolean equals(Object other) {
      if(other instanceOf Ghost) {
         Ghost g = (Ghost)other; // zombie
         return
      } else return false;
   }

}
```

7. **Write code to create sprites based on the position data in a file.**

```
public class Game {
   public static void main(String[] ) {
```