i) Creating Sublists. ii) Writing an efficient list implementation
iii) Object class has *toString* and *equals* method
iv) Shallow and Deep Copies review
MP5 re-graded tonight. Midterm II is next Wed.

```
class RobotAction {
// Class variable(s):
   public static final String[] JOINTS= {"Base","Arm","Elbow","Wrist","Clamp"};
// Instance variable(s)
   private double angle;
   private int index;              // joint index (see JOINTS)
// Getters and Setters:
   public int getIndex() { return this.index; }
   public double getAngle() { return this.angle; }

   /** Returns the jointname, a hypen, followed by the angle.*/
   public String toString() {



   }
   /** Returns true index is a valid index of JOINTS array
   public boolean isValid() {



   }
 //#3  Write a 'copy' method to create and initialize a new RobotAction object
which is identical to this one.
   public RobotAction copy() {




   }
//#4 Actually, you know about Copy Constructors. So write a copy
constructor version:
```

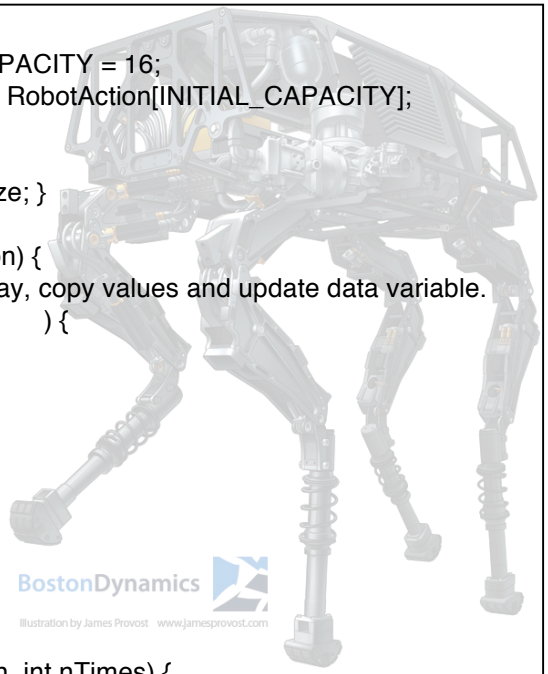
```
class ActionList {
  private static final int INITIAL_CAPACITY = 16;
  private RobotAction[] data  = new RobotAction[INITIAL_CAPACITY];
  private int size =0;

  public int getSize() { return this.size; }

  public void add(RobotAction action) {
     // no space=> Create a new array, copy values and update data variable.
     if (                                          ) {



    }
    data[size] = action;
 }
 public void add(RobotAction action, int nTimes) {
   for ( ; nTimes > 0 ; nTimes --) add(action);
 }
```



#7 Write an instance method *allActionsOK* that returns true only if all RobotActions in this list are valid.

#8 Write an instance method *getInvalidActions* that takes no parameters returns a new ActionList. This list will only contain actions that are invalid. Use a shallow copy.

#9 Write an instance method *getActionsForJoint* that takes an integer 'j' - the joint index and returns a new list. This list will only contain actions for the specified joint index. Use a deep copy.

Assume you have a spot of paint with x,y &Color. Complete the *equals* method

```
class Spot {
   private int x, y;
   private Color color;
   public boolean equals(Object other) {

         Spot spot2 = (Spot) other;

      return
   }
….
}
```

Complete the following SpotList class to create an equals method

```
class SpotList{
  private Spot[] array; // Assume entries 0 upto count -1 are non-null Spots
  private int count;

  public boolean equals(Object other) {

         SpotList list2= (SpotList) other;
```