```java

/**
 * This class Represent a Musical note.
 *
 * <ul>
 * <li> Name: Note.java
 * <li> Description: Note
 * <li> Class: Java 145
 * <li> Instructor: Ken Hang
 * <li> Date: Feb 4 2015
 * </ul>
 *
 * @author  Hai H Nguyen (Bill)
 * @version Winter 2015
 */

public class Note {

   private double duration;

   private Pitch note;

   private int octave;

   private Accidental accidental;

   private boolean repeat;

   /**
    * Main Constructor of Note
    * @param duration      Duration of Node, must be positive
    * @param note       Pitch of the Node
    * @param octave     The octave must be within [1,9]
    * @param accidental Indicator to Raise or Lower note
    * @param repeat     Repetition indicator
    */
   public Note(double duration, Pitch note, int octave,
            Accidental accidental, boolean repeat) {
      this (duration, note, repeat);

      if (octave >= 10 || octave <= 0){
         throw new IllegalArgumentException ("Invalid Octave (0,10): " + octave);
      } else {
         this.octave = octave;
      }

      this.accidental = accidental;
   }

   /**
    * Short constructor of Note
    * Initialize the note with passed duration, pitch and repeat indicator
    * @param duration      Duration of Node, must be positive
    * @param note       Pitch of the Node
    * @param repeat     Repetition indicator
    */
   public Note(double duration, Pitch note, boolean repeat) {
      setDuration(duration);

      this.note = note;

      this.repeat = repeat;
   }

   /**
    * Get the duration of the note.
    * @return        Return the Duration of the Node
    */
   public double getDuration() {
      return duration;
   }

   /**
    * Set the duration of the note to time.
    * @param duration      New Duration of the Note
    */
   public void setDuration(double duration) {
      if (duration < 0){
         throw new IllegalArgumentException ("Invalid Duration (0,oo): " + duration );
      } else {
         this.duration = duration;
      }
   }

   /**
    * Tell if the note is the indicator of a repeated section
    * @return        The repeat Indicator of the Note
    */
   public boolean isRepeat() {
      return repeat;
   }

   /**
    * Play the sound this note represents.
    */
   public void play() {
      StdAudio.play(duration, note, octave, accidental);
   }

   /**
    * Returns a string represent the note in the format:
    * If rest: "<duration> <pitch> <repeat>"
    * Else: "<duration> <pitch> <octave> <accidental> <repeat>"
    * @return        A formatted string describe the note
    */
   public String toString() {
      String out = duration + " " + note.toString() + " ";

      if (!note.equals(Pitch.R)){
         out += octave + " " + accidental.toString() + " ";
      }

      out +=  (repeat ? ("true"):("false"));

      return out;
   }
}
```