CS435/Spring 2014/Handout: LZW program

Encoder

```
import java.util.*;

public class LZWEncoder extends Stack<Stack<Integer>>{
    public static final int CLEAR_TABLE=256;
    public static final int END_OF_DATA=257;
    private Stack<Integer> match;

    LZWEncoder(){super();reset();}
    private void reset(){
        clear();
        match=new Stack<Integer>();
        for(int i=0; i<256; i++) {
            Stack<Integer> s=new Stack<Integer>();
            s.push(i);
            push(s);
        }
        push(new Stack<Integer>()); // 256
        push(new Stack<Integer>()); // 257
    }
    public void encode(){
        match=new Stack<Integer>();
        emit(CLEAR_TABLE);
        try{
            int n;
            while((n=System.in.read())!=-1){
                encode(n);
            }
            emit(END_OF_DATA);
        }catch(Exception e){
            e.printStackTrace();
        }
    }
    private boolean isFull(){return size()==4096;}
    private int code(Stack<Integer> s){return indexOf(s);}
    private void emit(int n){System.out.println("emit "+n);}
    private static Stack<Integer> copy(Stack<Integer> s){
        Stack<Integer> r=new Stack<Integer>();
        for(int i=0; i<s.size(); i++) r.push(s.elementAt(i));
        return r;
    }
    private void encode(int n){
        System.out.println("encode "+n);
```

```
            System.out.println("match "+match);
            if(isFull()){
                emit(code(match));
                emit(CLEAR_TABLE);
                reset();
                match.push(n);
            } else {
                match.push(n);
                if(!contains(match)){
                    push(copy(match));
                    System.out.println((size()-1)+" "+peek());
                    match.pop();
                    emit(code(match));
                    match.clear();
                    match.push(n);
                }
            }

        }
    }
```

Input

```
-----A---B
```

Output

```
emit 256
encode 45
match []
encode 45
match [45]
258 [45, 45]
emit 45
encode 45
match [45]
encode 45
match [45, 45]
259 [45, 45, 45]
emit 258
encode 45
match [45]
encode 65
match [45, 45]
260 [45, 45, 65]
emit 258
encode 45
match [65]
261 [65, 45]
emit 65
encode 45
match [45]
encode 45
match [45, 45]
encode 66
match [45, 45, 45]
262 [45, 45, 45, 66]
emit 259
encode 10
match [66]
263 [66, 10]
emit 66
emit 257
```