import java.io.\*;

import java.util.\*;

//import java.lang.\*;

public class SimpleShell

{

public static void main(String[] args) throws java.io.IOException {

String commandLine;

int index = 0;

BufferedReader console = new BufferedReader //make buffered Reader for this shell

(new InputStreamReader(System.in));

ArrayList<String> total = new ArrayList<String>();

ProcessBuilder pb = new ProcessBuilder();

File firadd = new File(System.getProperty("user.home")); //Initialize address of workspace

pb.directory(firadd); //change workspace to /home/sean

while (true) {

// read what the user entered

System.out.print("jsh>");

commandLine = console.readLine();//on here, program get what user typed

String[] words = commandLine.split("\\s");//split the user input space by space

ArrayList<String> list = new ArrayList<String>();// make list to contain those words

for(int i=0;i<words.length;i++) { //put words to list in order

list.add(words[i]);

}

total.add(commandLine); //store every list into total,to use later for history function

//System.out.println(list);

try{

// if the user entered 'exit' or 'quit', kill this program

if(list.contains("exit")|list.contains("quit")) {

System.out.println("Goodbye.");

System.exit(0);// end this program here

}

// if the user entered nothing, just loop again

if (commandLine.equals("")) {

total.remove(""); //remove empty space from history

System.out.println("Hey, please type something!"); //to warning the lazy user ^^

continue;

}

/\*if(list.contains("ps")) { //activate here when you run in windows

list.set(2,"tasklist");

}

if(list.contains("ls")) {

list.set(2,"dir");

}

if(list.contains("cat")) {

list.set(2,"type");

}\*/

//'history !!' will run in here

if(list.get(list.size()-1).equals("!!")){ //sort out input with "!!"

String[] again = total.get(total.size()-2).split("\\s"); //split last input to make it understandable words

ArrayList<String> AG = new ArrayList<String>();

for(int i=0;i<again.length;i++) {

AG.add(again[i]);

}//make Arraylist of understandable words

pb.command(AG); //run that words

total.remove("!!"); //remove !! from history

}//!<integer value i> command

else if(list.get(list.size()-1).charAt(0) == '!'){

int b = 0;

String str = list.get(list.size()-1);

String num = str.replace("!","");// sort out "!"

b =Integer.parseInt(num);//get number value from input

if(b<=total.size()) {//check if integer entered isn't bigger than history size

String[] recall = total.get(b).split("\\s"); //split that specific input to make it understandable words

ArrayList<String> RC = new ArrayList<String>();

for(int i=0;i<recall.length;i++) {

RC.add(recall[i]);

}//make Arraylist of understandable words

pb.command(RC); //run that words

}

total.remove("!"+b); //remove !b from history

} else {

pb.command(list); // run other cases

}

//cd test

if(list.contains("cd")){

if(list.get(list.size()-1).equals("cd")){//when user type only 'cd'

File home = new File(System.getProperty("user.home"));//file to make directory goes back to first place

pb.directory(home);

continue;

}else if(list.get(list.size()-1).contentEquals("..")) {// when user type 'cd ..'

File back = new File(pb.directory().getParent()); // file address goes up once

pb.directory(back);

continue;

}else{ //when user type cd + something

String dir = list.get(1);

File newPath = new File(dir);

System.out.println(newPath);

boolean exists = newPath.exists();//check whether that address is exist or not

if(exists){//if that address was real

System.out.println("New directory=>" + dir); //added the "New directory=>"

pb.directory(newPath);//change working directory to newpath

continue;

} else{

if(dir.contains(pb.directory().getName())) { //if user type repeated address

if(dir.contains("/")) {

String find = new File(pb.directory().getParent()+newPath).getAbsolutePath(); //address goes up once + add newPath

File as = new File(find); //change address to that address

pb.directory(as);//now type the address of that user input address

System.out.println(as);

continue;

}

else {

File find2 = new File(pb.directory().getParent()); //address goes up once

pb.directory(find2);

String NEW1 = new File(pb.directory()+"/"+dir).getAbsolutePath(); //if user type relative path, add parent address and "/"

File N1 = new File(NEW1);

pb.directory(N1);

continue;

}

}

String NEW = new File(pb.directory()+"/"+dir).getAbsolutePath(); //if user type relative path, add parent address and "/"

File N2 = new File(NEW);

boolean Rexists = N2.exists(); //check whether that relative path is exist or not

if(Rexists) {

pb.directory(N2);

System.out.println(N2);

continue;

}

System.out.print("Path Error!\n"); //print when those addresses are not exist

continue;

}

}

} //cd ends

//display history of shell with index

if(list.get(list.size()-1).equals("history")){//check whether user want to see history

for(String s : total) //put data in 'total' to string s

System.out.println((index++) + " " +s); //print out the history with number

index = 0;

pb.command();

continue;

}

if(list.get(list.size()-1).contains("history")){

int h = 0;

String str = list.get(list.size()-1);

String num = str.replace("history","");// sort out "history"

h =Integer.parseInt(num);//get number value from input

if(h<total.size()&&h>=0) {//check if integer entered isn't bigger than history size

System.out.println(h +" "+total.get(h));

}else {

System.out.println("History not existed");

}

pb.command();

continue;

}

//System.out.println(total);

Process process = pb.start(); // start process builder

//obtain the input stream

InputStream is = process.getInputStream();

InputStreamReader isr = new InputStreamReader(is);

BufferedReader br = new BufferedReader(isr); //store data in buffer

//read the output of the process

String line;

while ( (line = br.readLine()) != null) //printout every output results

System.out.println(line);

br.close();

System.out.println(list);//to show to user what they typed

}//try

//catch ioexception, output appropriate message

catch (IOException e){

System.out.println("Input Error, Please try again!");

/\*\* The steps are:

(1) parse the input to obtain the command and

any parameters

(2) create a ProcessBuilder object

(3) start the process

(4) obtain the output stream

(5) output the contents returned by the command \*/

}//catch

catch(NumberFormatException n) {

System.out.println("Input Error, Please try again!!!");

}

}//while

}//main

}//class