Question: https://leetcode.com/problems/basic-calculator/

So the idea is to maintain a sum until we face a opening parenthesis, once we encounter a opening parenthesis we store the previous sum and the sign before the bracket to the stack, and reset res and sign. So that we can maintain a fresh calculation for the expression inside the bracket.

And once we encounter a closing parenthesis, we multiply the result of the expression from inside the parenthesis to the sign stored in stack and add the last sum encountered before encountering the opening parenthesis.

Code:  
class Solution {

public int calculate(String s) {

int sign=1, res=0;

Stack<Integer> stck = new Stack<Integer>();

s=s.trim();//remove leading and triling spaces

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

char c= s.charAt(i);

if(c=='-')sign=-1;

else if(c=='+') sign=1;

else if(Character.isDigit(c)){

int n=c-'0';

while(i+1<s.length() && Character.isDigit(s.charAt(i+1))){

n=n\*10+(s.charAt(i+1)-'0');

i++;

}

res+=(n\*sign);

}

else if(c == '('){

stck.push(res);

stck.push(sign);

res=0;

sign=1;

}

else if(c == ')'){

res=res\*stck.pop()+stck.pop();

}

}

return res;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>