CODE SMELL TIPS

1. **La nivel de APLICATIE:**
   - **Cod duplicat**: cod identic sau foarte similar există în mai multe locuri.
   - **Complexitate agravată**: utilizarea forțată de design patternuri complicate acolo unde nu este cazul.
   - **NO magic numbers**.
   - **Variabile temporare** redundant/inutil create.

2. **La nivel de CLASA:**
   - Class names begin with an upper case letter [A-Z]
   - **Clasă mare**: o clasă care a crescut prea mult.
   - **Feature envy**: o clasă ce folosește excesiv metode ale unei alte clase.
   - **Intimitate nepotrivită**: o clasă care are dependențe de detaliile de implementare ale unei alte clase.
   - **Refused bequest**: o clasă ce suprascrie o metodă a clasei de bază în așa fel încât contractul clasei de bază nu mai este respectat de clasa derivată.
   - **Lazy class / freeloader**: clasă care face prea puțin.
   - **Abuz de literali**: în loc de literali ar trebui folosite în principal constante cu nume, pentru a ameliora lizibilitatea și pentru a evita erorile. Literalii ar trebui externalizați în fișiere de resurse, pentru a facilita localizarea software-ului dacă se dorește instalarea sa în regiuni sau contexte lingvistice diferite.
   - **Complexitate ciclomatică**: prea multe bucle și ramificări; aceasta poate indica faptul că o funcție trebuie divizată în funcții mai mici, sau că are potențial de simplificare.
   - **Downcasting**: un cast care rupe modelul de abstracție; abstracția poate fi refactorizată sau eliminată.
   - Do not use multiple inheritance of implementation. You must use single inheritance for implementation and can use multiple inheritance for interfaces.
   - Use the keyword override after the overridden virtual function declaration; don't use virtual again because it is redundant and confusing
   - Use the final keyword to seal a virtual method instead of pre-C++11 hacks

3. **La nivel de METODA:**

- **Prea mulți parametri**: o listă lungă de parametri este greu de citit și complică apelarea și testarea metodei. Poate sugera că funcția este slab concepută și că codul ar trebui refactorizat pentru a atribui responsabilitățile într-o manieră mai limpede.
- **Metodă lungă**: o metodă, funcție sau procedură ce a devenit prea mare.
- A function should be written to perform one task. Complex tasks should be broken up into many simpler functions for better readability and understanding. Smaller specialized functions can be reused by other functions requiring similar processing.
- Limit function length to 50 lines of source code, whenever practical.
- **Identificatori excesiv de lungi**: în particular, utilizarea convențiilor de denumire pentru dezambiguizare ar trebui să fie ceva implicit în arhitectura software.
- **Identificatori excesiv de scurți**: numele unei variabile trebuie să reflecte funcționalitatea ei dacă nu este ceva evident; de exemplu, variabilele de o singură literă sunt frecvente în manuale și în exemplele didactice și devin o practică încetățenită la începători, dar în proiectele practice rolul lor nu mai este înțeles atunci când codul este reanalizat în procesul de mentenanță – representative, fara abrevieri
- **Retur excesiv** de date: o funcție sau metodă care returnează mai mult decât are nevoie codul apelant.
- **Const correctness (https://www.youtube.com/watch?v=ZWRRXP-XFvY)**

WHAT ABOUT?