1. İsimlendirme kurallarına uyulmaması. (bkz. Naming convetions, Style guidlines)
2. Yorum ve açıklama yazılmaması. (bkz. Comments)
3. Büyük sınıflar. Bir sınıf bir nesneyi betimlemeli, sadece bir sorumluluğu olamalıdır. Farklı işler yapmamalıdır. (bkz. Single responsibility principle)
4. Veri saklayan sınıflar. Herhangi bir işi yoktur. Çok fazla alan (fields) içerir. (bkz. Data class)
5. Bir sınıfın işiyle ilgili çok az şey yapması. (bkz. Lazy class)
6. Sınıf içerisinde belli durumlarda kullanılan alanlar. (bkz. Temporary field)
7. Alt sınıf taban sınıfın veri ve methodlarının hepsini kullanmıyor. (bkz. Refused bequest)
8. Bir sınıfta değişiklik yapıldığında, sınıfın ilgisiz yerlerinde de değişiklik yapılması. (bkz. Divergent change)
9. Bir sınıfta değişiklik yapıldığında, birçok farklı sınıfta da değişiklik yapılması. (bkz. Shotgun surgery)
10. Bir sınıfın başka bir sınıfın üyelerini aşırı kullanması. (bkz. Feature envy)
11. Bir sınıfın tüm işlerini başka sınıflara yaptırması. (bkz. Middle man)
12. Message chains

A a;

a.b().c().d().e().f().g().h().i();

1. Inappropriate Intimacy.
2. Primitive Obsession.
3. Data Clumps.
4. Alternative Classes with Different Interfaces.
5. Parallel Inheritance Hierarchies.
6. Incomplete Library Class.
7. Speculative Generality.
8. Combinatorial Explosion.
9. Indecent Exposure.
10. Downcasting.
11. Cyclomatic complexity.
12. Üst sınıf, alt sınıfa bağlı.
13. Uzun fonksiyon. Her fonksiyon sadece tek bir iş yapmalıdır. Fonksiyonların kısa olması okuma ve anlaşılmayı kolaylaştırır.
14. Methodların fazla parametreye sahip olması.
15. Çok sayıda kod bloğunun iç içe yuvalanması.

if (true)

{

if (true)

{

if (true)

{

if (true)

{

if (true)

{

// ...

}

}

}

}

}

1. Hardcoded sayı ve kelimeler. (bkz. Magic numbers)
2. Kod tekrarı (bkz. Duplicated code)
3. Kullanılmayan değişkenler.
4. Kullanılmayan method parametreleri.
5. Kullanılmayan methodlar, sınıflar. (bkz. Dead code)
6. “goto" deyimini kullanmak.

void f()

{

label:

// do something

goto label;

}

1. == (eşit operatörü) yerine = (atama operatörünü) kullanmak

if (i = 0)

{

// do something

}

1. “switch” deyiminin “break” keywordünü içermemesi.

switch (i)

{

case 1:

break;

case 2:

// do something

default:

break;

}

1. void pointer.

int n;

float f;

void\* ptr;

ptr = &n; // valid

ptr = &f; // valid

1. Erişilemeyen kod.

int f()

{

// do something

return 1;

int i;

}

1. Macro yerine inline function, enum, const kullanılmalı.
2. Boş kod blokları. (if, else, for, while, do/while, switch, try, catch, finally)

if (condition)

{

}

if (condition)

{

// do something

}

else

{

}

for (int i = 0; i < length; i++)

{

}

1. Methodların içerisinde parametrelere tekrardan değer atamak.

void f(int i)

{

// do something

i = 0;

}

1. Yerel bir değişkenin referansını döndürmek.

int& f()

{

int i;

return i;

}

1. Yanlış “;” (noktalı virgül) kullanımı

if (true);

DoSomething();

while (true);

DoSomething();

for (size\_t i = 0; i < length; i++);

DoSomething();

int i;;

DoSomething();;

;

1. Referans veya işaretçilerin tip dönüşümleri.

class A

{

/\* ... \*/

};

class B

{

/\* ... \*/

};

A \* a = new A;

B \* b = reinterpret\_cast<B\*>(a);

class A

{

/\* ... \*/

};

class B : A

{

/\* ... \*/

};

A a;

B& b = dynamic\_cast<B&>(a);

1. Dinamik belleklerin serbest bırakılmaması, dosyaların kapatılmaması, vb.

int\* i;

//do something

//delete i;

fstream file;

file.open("test.txt");

//do something

//file.close();

1. Exceptions (istisnai durum) yutulmamalı. Sadece işlenebilecekse yakalanmalı.
2. Spaghetti code.
3. Satırların uzun olması. Ekrana sığmaması. Okumayı zorlaştırır.
4. Değişkenler, fonksiyonlar, sınıflar anlamlı isimlere sahip olmalı.
5. Sonsuz döngü.