**Aula Prática 12**

1) Implemente o atributo UnsafeType, aplicável múltiplas vezes a classes, com dois construtores: o primeiro recebe um tipo (representado pela respectiva instância de Type) considerado "não seguro" pela classe *target* do atributo; o segundo recebe o tipo e uma String indicando o nome de métodos “não seguros” desse tipo.

2) Implemente o método estático bool IsUnSafeCallback(Type t, Delegate d) da classe ReflectionUtils. O método retorna true se o *método associado ao delegate* representar um método “não seguro” do tipo especificado pelos atributos UnsafeType aplicados ao tipo representado por t.

3) Sejam os tipos:

interface IAlarm { event EventHandler trigger; }

class AlarmProcessor1 {

public static void OnAlarmX(object src, EventArgs arg){

...}

... }

class UnsafeTypeException : Exception {

private Type unsafeType; private String method;

public UnsafeTypeException(Type t, String met) {

unsafeType = t; method = met; }

public override string ToString() {

return base.ToString() + "unsafeType: " + unsafeType + " - Method: " +

method + "\n"; }

}

Tirando partido do código pedido nas duas alíneas anteriores, implemente a classe AlarmGenerator, que implementa a interface IAlarm e aceita no evento trigger *callbacks* de qualquer tipo, com excepção do *callback* AlarmProcessor1.OnAlarmX. Na tentativa de registar este *callback* é lançada uma excepção do tipo UnsafeTypeException