Estándares de codificación

Musify

Cruz Portilla Mauricio

Romero Peña Arturo Iván

Ingeniería de Software

# Estándar de codificación C#

|  |  |
| --- | --- |
| Encabezados en clases, métodos o atributos | Para las clases y métodos, dar una explicación breve que resulte necesaria para comprender su propósito, indicándolo también para cada uno de sus parámetros y su retorno (si es diferente a *void*).  Para el caso de los atributos, si es necesario, dar una explicación muy breve sobre su utilidad, y si se trata listas o diccionarios, agregar a qué corresponden sus valores. |
| Formato de encabezado | Para clases y atributos:  <summary>  Descripción  </summary>  Para métodos:  <summary>  Descripción  </summary>  <param name=”param1”>Descripción</param>  <returns>Descripción</returns> |
| Comentarios en código | En caso de ser extremadamente necesario, agregar máximo 2 comentarios muy breves en código de métodos para dar una guía de a qué se refiere algo en concreto. Cada comentario debe ir justo después del fin de la línea, delimitado por un punto y coma. |
| Buen comentario | EngineNetwork.DoNetworkOperation(onExecute: () => {  Thread.Sleep(2000); // allows session thread to load latest game data  return true;  }, onSuccess: () => {  // code skipped  }, onFinish: null, null, false); |
| Mal comentario | EngineNetwork.DoNetworkOperation(onExecute: () => {  // allows session thread to load latest game data because otherwise it wouldn’t pass correctly the right data to the newest window, causing an ugly error.  Thread.Sleep(2000);  return true;  }, onSuccess: () => {  // code skipped  }, onFinish: null, null, false); |
| Nombramiento de clases, variables, atributos, constantes, métodos y parámetros | * El nombre de una clase siempre debe comenzar por una letra mayúscula cada palabra, y debe usar un nombre nominal como *Account*¸ *Player, GameCategory*. * El nombre de un atributo privado, interno o protegido debe comenzar por una letra minúscula y las demás palabras con empezar con letra mayúscula, como *timeRemaining*. * El nombre de un atributo público debe comenzar cada palabra que lo conforma por una letra mayúscula. * El nombre de una constante, caracterizada también por ser *readonly*, debe ser escrito completamente en mayúsculas, y cada palabra separada por un guión bajo. * El nombre de un método siempre debe comenzar cada una de sus palabras por una letra mayúscula. * El nombre de un parámetro debe comenzar por una letra minúscula, y cada una de sus demás palabras por una letra mayúscula. Deben usar un nombre nominal como *serviceAccount*, *gameList*. |
| Formato de código utilizando operadores, palabras reservadas para estructuras de control y/o llaves | * Los operadores deben tener un espacio antes y después. * Las palabras reservadas para estructuras de control deben tener un espacio después, antes de abrir su respectivo paréntesis. * Para todo caso ajeno a la definición o llamada de una función, los paréntesis que abren deben tener un espacio antes; los paréntesis que cierran deben tener un espacio después (este último es solo si el siguiente carácter no es un punto y coma). * Las llaves que abren deben tener un espacio antes y un salto de línea después, y las que cierran deben tener un salto de línea antes. Para el caso de atributos de clases, se puede tener ambas llaves en la misma línea. * Las estructuras de control tienen la llave que abre en la misma línea donde se define, mientras que la que cierra está en una línea más abajo. Para el caso de estructuras de control con más de un bloque (*try-catch-finally, if-else*), la definición de los siguientes bloques se realiza justo después del cierre del bloque anterior y se abre en la misma línea. * Cada línea debe tener un máximo de 150 caracteres. |
| Ejemplo de formato de código | public bool Reload() {  return EngineNetwork.EstablishChannel<IAccountService>((loginService) => {  var account = loginService.GetAccount(id);  if (account.Id == 0) {  return false;  }  username = account.Username;  password = account.Password;  email = account.Email;  coins = account.Coins;  creationDate = account.CreationDate;  isVerified = account.IsVerified;  validationCode = account.ValidationCode;  return true;  });  } |