Kodun okunabilir olması neden önemlidir?



www.yzokul.com.tr

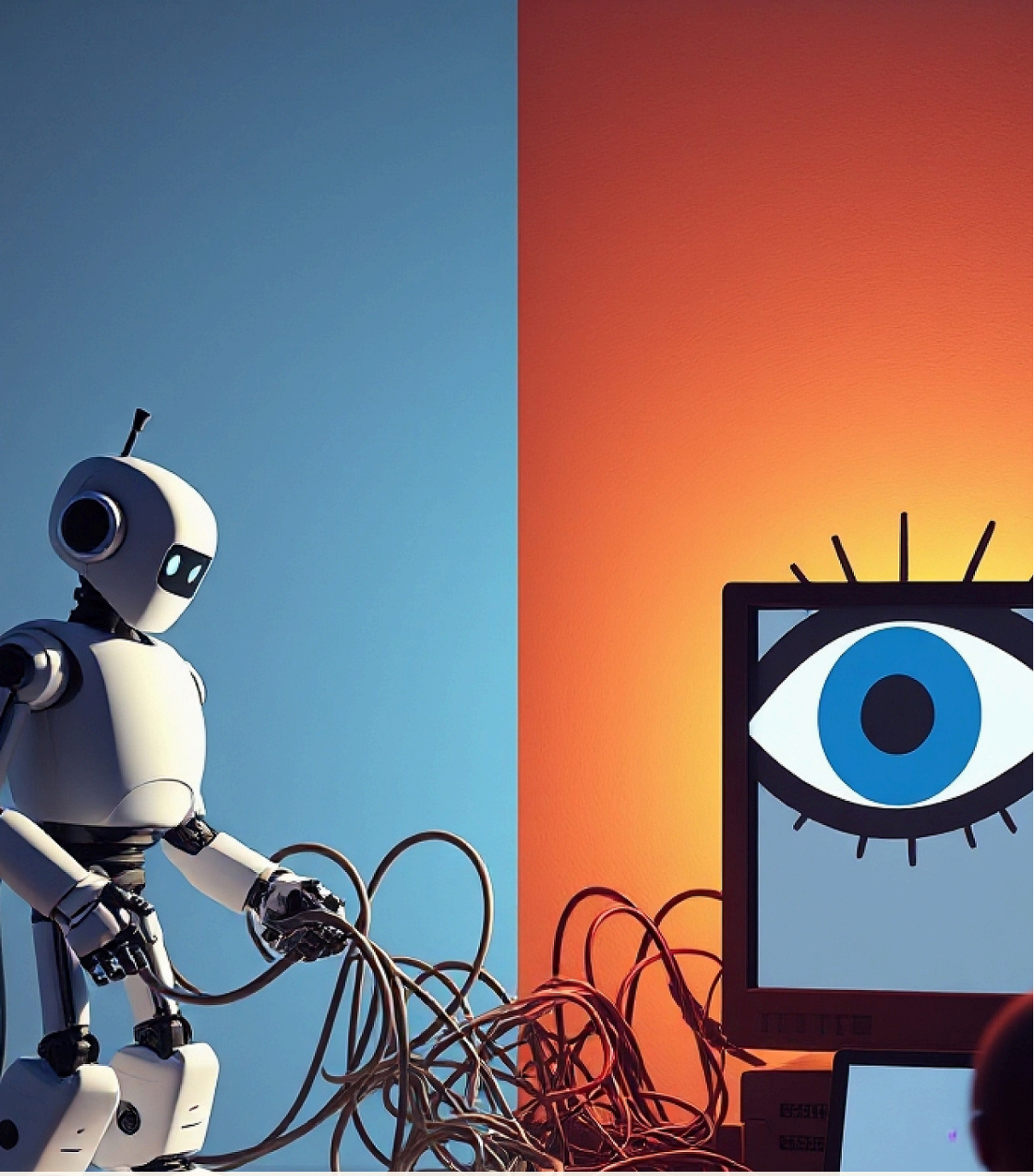
Kodun okunabilir olması neden önemlidir?



Kod, bilgisayar için yazılmış talimatlardır. Ama insanlar da onu okumak zorundadır ve bunu yazdıklarından çok daha sık yaparlar. İşte bu yüzden okunabilir kod önemlidir: yalnızca makinelerin çalıştırması için değil, insanların anlaması için de kolay olmalıdır. Kısa yanıt: açık kod, onu sonradan okuyan herkes için zaman kazandırır ve hataları önler.



Bir program bir kez yazılır ama birçok kez okunur. Kendi kodunuzu haftalar sonra yeniden okursunuz ve başkaları da okur. Kodun takip edilmesi kolaysa her okuma daha hızlıdır. Kodu okunabilir kılmak, birisi dosyayı her açıp ne yaptığını anlamaya çalıştığında tekrar tekrar karşılığını veren bir yatırımdır.



Bir bilgisayar dağınık kodu şikâyet etmeden çalıştırabilir. Talimatlar geçerli olduğu sürece adları ya da sırayı önemsemez. İnsan farklıdır: insan okuyucu mantığı takip edebilmelidir. Kodun iki hedef kitlesi vardır: onu çalıştıran makine ve onu okuyan insanlar. Okunabilir kod her ikisine de hizmet eder ama insan düşünülerek yazılır.



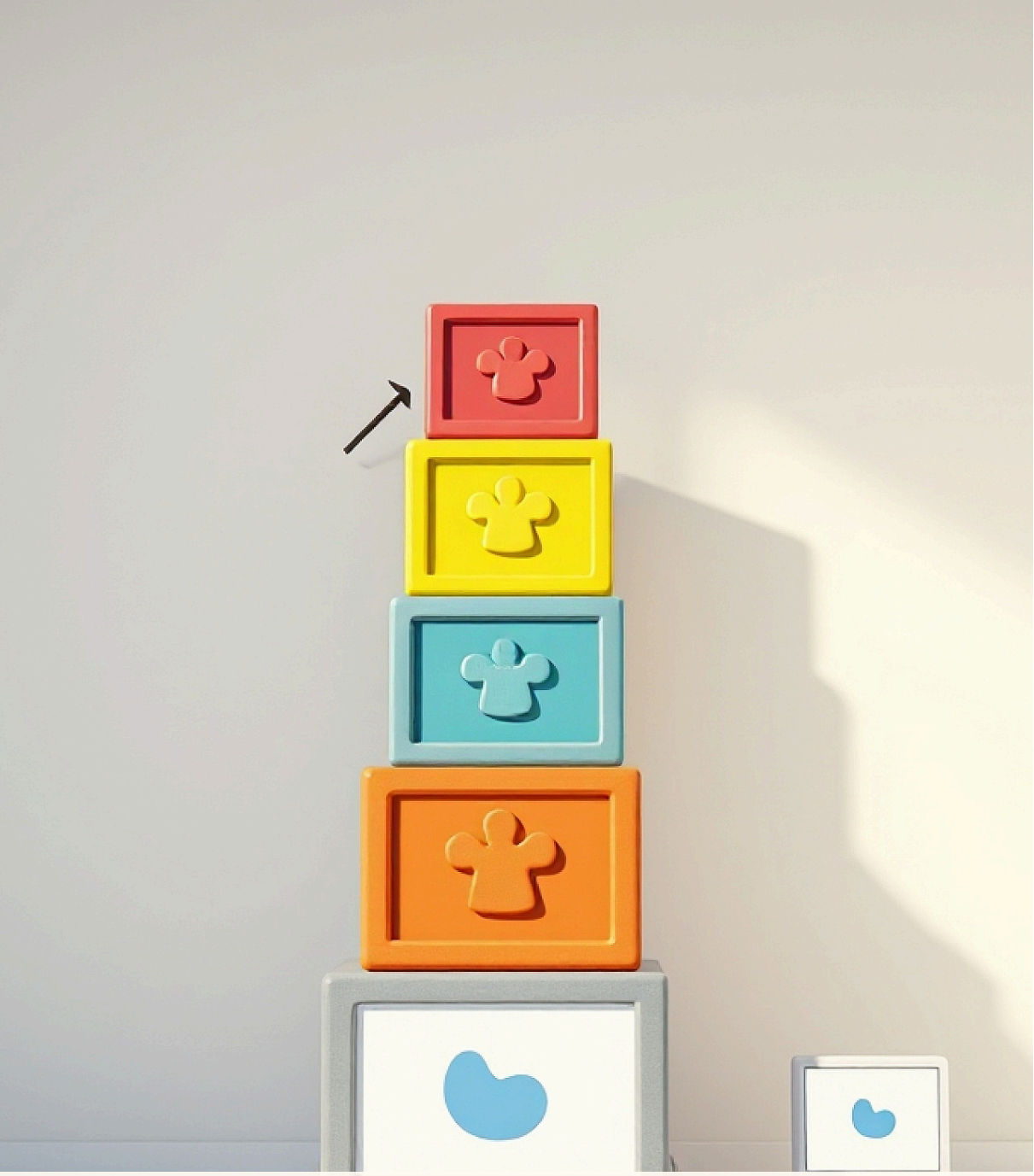
Okunabilir kod, bir insanın tahmin etmeden hızlıca anlayabildiği koddur. Her parçanın ne yaptığı ve neden orada olduğu açıktır. Amacı bilmek için her satırı izlemeniz gerekmez. Bir bölümün önünde durup kafa yormanız gerekiyorsa, o bölüm yeterince okunabilir değildir. Amaç, bir bakışta anlamaktır.



Adlar, programdaki şeylerin, örneğin değerlerin, listelerin veya fonksiyonların etiketleridir. İyi bir ad, neyi tuttuğunu söyler, böylece hatırlamak ya da tahmin etmek zorunda kalmazsınız. total_price gibi bir ad, t gibi bir addan daha fazlasını söyler. İçeriği tanımlayan adlar seçin, kod kendini açıklar hale gelsin. Anlamlı adlar, okunabilirlik için en basit kazanımlardan biridir.



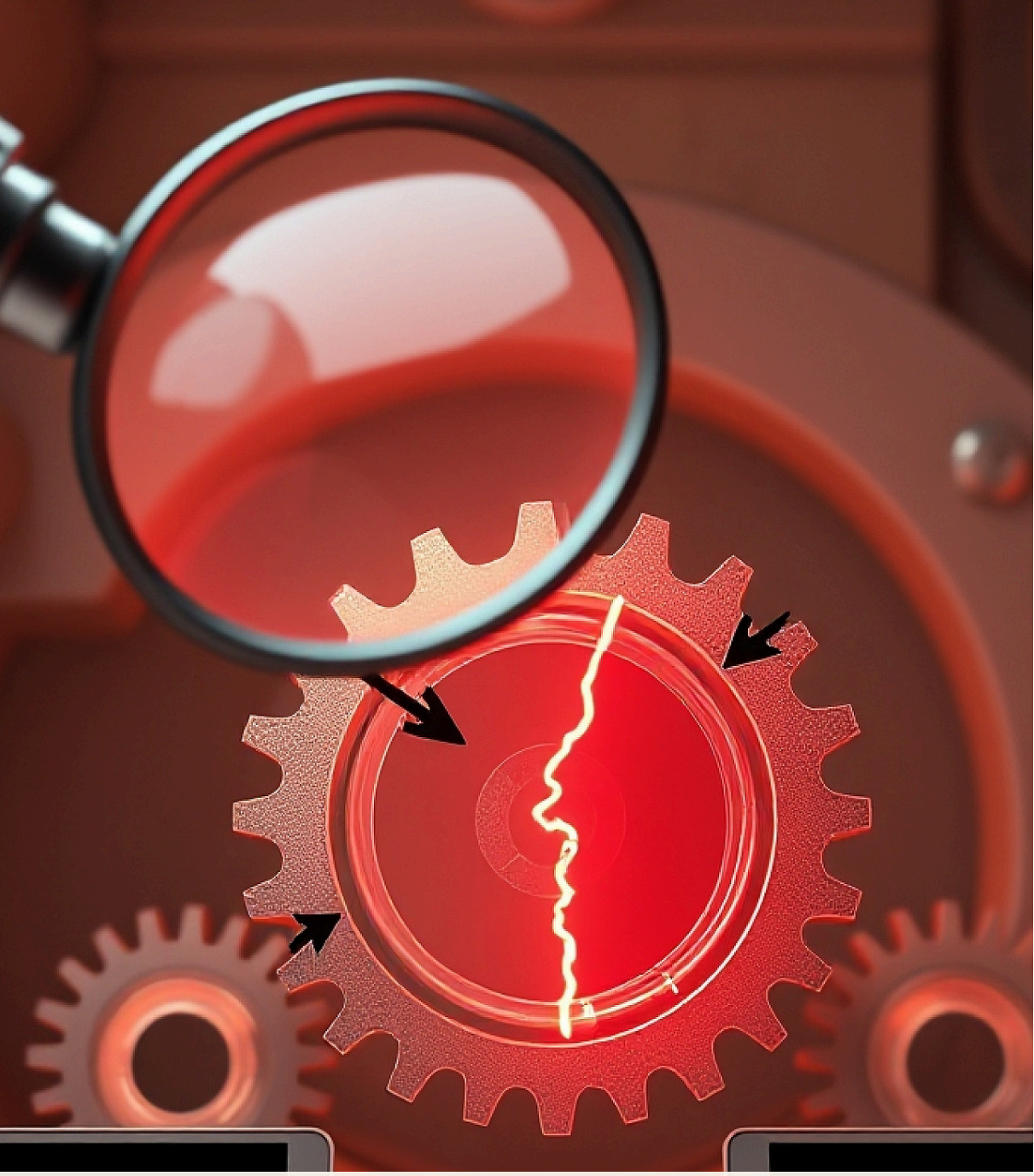
İki adı karşılaştırın: x ve ogrenci_sayisi. İkincisi açıkça öğrenci sayısı anlamına gelir, oysa x neyi tuttuğu hakkında hiçbir şey söylemez. Çoğu durumda tanımlayıcı ad kazanır. Birkaç harf daha fazla yazabilirsiniz ama her okuyucu zaman kazanır. Kısa adları yalnızca döngü sayacı gibi küçük, apaçık durumlarda kullanın.



Kodu açık ve mantıklı bir sıraya koyun ki okuyucu adım adım takip edebilsin. İlgili satırları bir arada gruplayın ve akışı yukarıdan aşağıya koruyun. Düzenli bir yapı, programda kaybolmadan ilerlemenizi sağlar. Sıra karışıksa doğru kod bile takip edilmesi zordur. Düzenli bir yerleşim, okunabilirliğin bir parçasıdır.



Küçük fonksiyonlar, tek bir dev bloktan daha kolay okunur. Her küçük fonksiyon tek ve açık bir iş yapmalıdır ve her parçayı kendi başına anlayabilirsiniz. Bir fonksiyon çok uzarsa, onu tanımlayıcı adlara sahip daha küçük parçalara bölün. Bu, tüm programı kavramayı kolaylaştırır, çünkü her şeyi aklınızda tutmadan bir seferde bir parçayı okuyabilirsiniz.



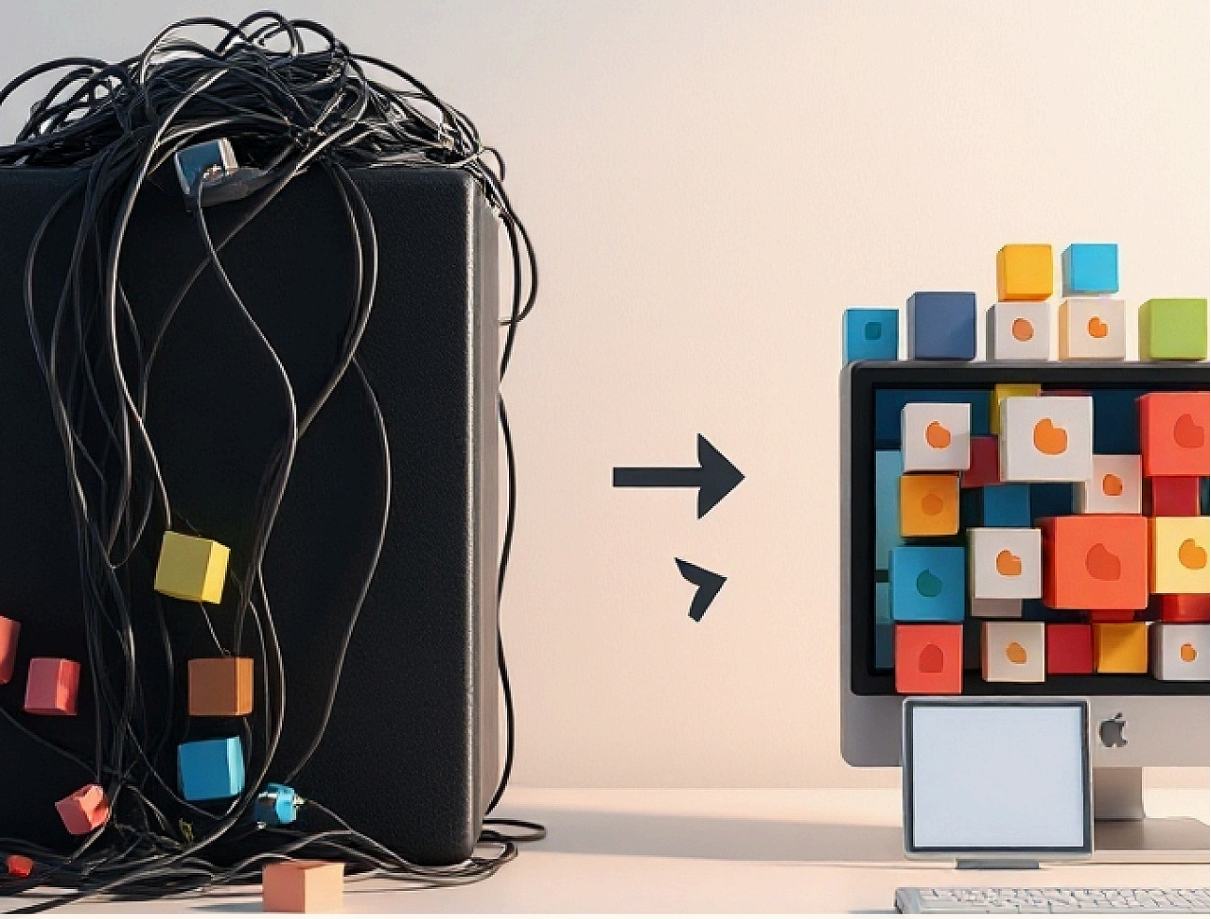
Kod açık olduğunda bir hatayı bulmak daha hızlıdır. Kafa karıştıran satırların arasında kazmak yerine işlerin nerede ters gittiğini görürsünüz. Okunabilir adlar ve yapı sizi olası soruna yönlendirir. Hata ayıklama, hataları bulup düzeltmek demektir ve kod kendini açıkladığında daha az zaman alır. Açık kod, sinir bozucu bir arayışı hızlı bir kontrole dönüştürür.



Mevcut kod anlaşılır olduğunda yeni bir özellik eklemek daha kolaydır. Yeni parçanın nereye uyduğunu ve neyi etkileyeceğini bilirsiniz. Okunabilir kod bir harita gibi davranır ve size değişiklik yapmak için güvenli yerleri gösterir. Kod karmaşık bir yumaksa her değişiklik riskli hissettirir. Açık kod, programı genişletmeyi normal ve sakin bir iş haline getirir.



Yaygın bir hata, her yerde a, b veya x gibi kısa, anlamsız adlar kullanmaktır. Bu adlar okuyucuları her değerin ne anlama geldiğini tahmin etmeye zorlar. Bugün hatırlayabilirsiniz ama gelecek hafta hatırlamayacaksınız. Anlaşılmaz adları student_count veya total_price gibi tanımlayıcı olanlarla değiştirin. Fazladan yazılan, her okuyucunun kazandığı zamana kıyasla küçüktür.



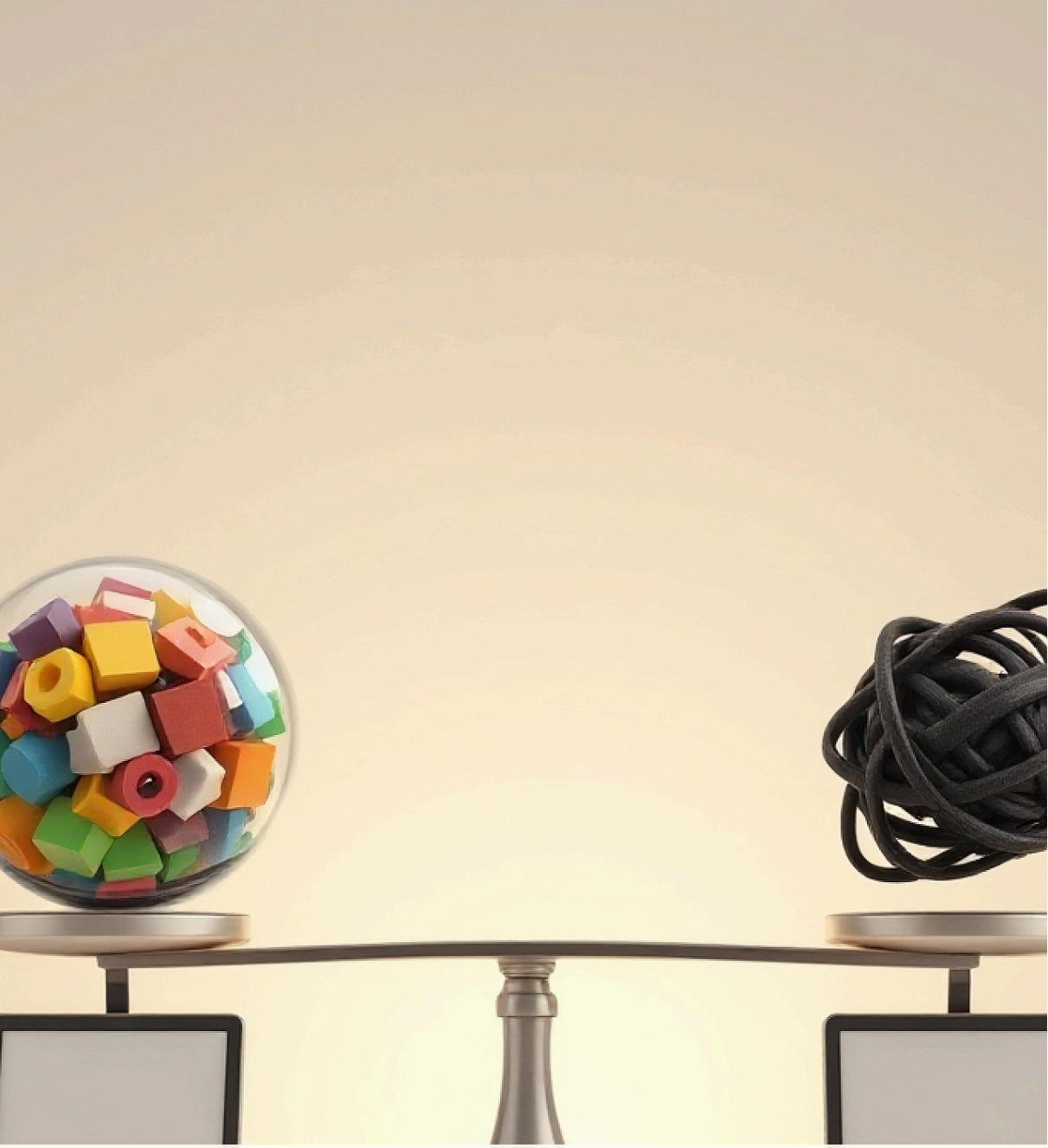
Bir başka yaygın hata, her şeyi tek bir uzun, karmaşık kod bloğuna sıkıştırmaktır. Dev bir bloğu takip etmek zordur çünkü parçaları göremezsiniz. Onu her biri tek bir amaca sahip küçük, açık fonksiyonlara veya bölümlere ayırın. Bu, programı okumayı ve test etmeyi çok daha kolaylaştırır. Küçük parçaları anlamak, tek bir büyük düğümden daha basittir.



Karışık, düzensiz kod çalışsa bile takip edilmesi zordur. Satırlar sırasız olduğunda ya da mantıksızca karıştığında okuyucu zihinsel bir harita kuramaz. İlgili şeyleri bir arada tutun ve açık bir sıra izleyin. Öngörülebilir bir sıra, şeylerin nerede olduğunu hatırlamanıza yardımcı olur. Düzen, üslup meselesi değildir; programı anlaşılır kılmakla ilgilidir.



Basit bir kontrol deneyin: kodu birine açıklıyormuş gibi yüksek sesle okuyun. Takılıyorsanız ya da bir adın ne anlama geldiğini tahmin etmek zorunda kalıyorsanız, kodun daha açık adlara veya yapıya ihtiyacı var. Bu test, kafa karıştıran kısımları hızla ortaya çıkarır. Bir dinleyiciye ihtiyacınız yok; kendi açıklamanızı duymak bile kodun nerede belirsiz olduğunu gösterir. O noktaları düzeltin ve tekrarlayın.



Okunabilir kod, yalnızca bilgisayarlar tarafından çalıştırılmakla kalmaz, insanlar tarafından anlaşılır. Açık adlar, düzenli yapı ve küçük fonksiyonlar, kodu okuyan herkes için, gelecekteki kendiniz dahil, zaman kazandırır. Hata ayıklamaya daha az, özellik eklemeye daha çok zaman harcarsınız. Sonuçta okunabilirlik, kodunuzla sizden sonra çalışacak insanlara saygı duymakla ilgilidir.



Temel yanıtı unutmayın: kod, yazıldığından daha sık okunur, bu yüzden onu insanlar için açık hale getirin. x yerine ogrenci_sayisi gibi anlamlı adlar kullanın. Düzenli bir yapı koruyun ve küçük fonksiyonlar yazın. Okunabilir kodda hata ayıklamak daha kolaydır ve genişletmek daha kolaydır. Bu alışkanlıklar programlarınızı insanlar için anlaşılır kılar, asıl önemli olan da budur.