Bilişim Teknolojileri 5. Sınıf Sınava Hazırlık

Kısa Notlar

1. Algoritma Sıralaması Nasıl Olmalı?

Bir program her zaman belirli bir sırayı izler:

1. **Başla** (Tüm algoritmaların ilk adımı)
2. **Veriyi Al** (Örn: Ortalamayı kullanıcıdan al)
3. **Şartı Kontrol Et** (Eğer ortalama $\geq$ 50 ise)
4. **Sonuç Yazdır** (Evet ise "Geçti", değilse "Kaldı")
5. **Bitir** (İşlem sonlanır)

2. Sahnenin Tam Merkezi

Scratch sahnesi bir koordinat düzlemidir. Tam orta (başlangıç) noktası her zaman **X: 0** ve **Y: 0** olarak kabul edilir. X yatay ekseni, Y dikey ekseni temsil eder.

3. En Sağ Nokta (X Sınırı)

Yatay eksende (X ekseni) kuklanın gidebileceği mesafeler sınırlıdır. Ekranın en solu **X = -240**, ekranın en sağı **X = 240** değerini alır.

4. En Alt Nokta (Y Sınırı)

Dikey eksende (Y ekseni) kuklanın yukarı ve aşağı sınırları şöyledir: Ekranın en üstü **Y = 180**, ekranın en altı **Y = -180** değerini alır.

5. Koordinat Hesaplama Mantığı

Koordinat sorularında en çok kafa karıştıran kısım budur:

1. değiştir **Ne Demek?**
Elinizdeki mevcut sayının üzerine eklemek veya ondan çıkarmak demektir.
x konumunu 30 değiştir (Mevcut X: 50 ise, X = 80 olur)

2. yap **Ne Demek?**
Elinizdeki sayıyı tamamen unutup konumu direkt o yeni sayıya eşitlemektir (sabitlemektir).
x konumunu 100 yap (Mevcut X: 80 ise, X direkt 100 olur)

6. Görünüm Menüsü (Mor Bloklar)

Kuklanın sahnede nasıl göründüğüyle ilgili tüm işlemler bu gruptan yapılır. Boyut, renk efektleri, konuşma balonları ("Merhaba! de") ve kostüm değişimlerinin hepsi mor renkli **Görünüm** sekmesindedir.

7. Kod Blokları Neden Renkli?

Scratch arayüzünde kodların farklı renklerle kategorize edilmesinin en büyük sebebi **kullanıcıya kolaylık sağlamaktır.** Bu sayede blokların hangi kategoriye ait olduğunu hemen ayırt ederiz ve aradığımız komutu saniyeler içinde bulabiliriz.

8. Projeyi Nasıl Başlatırız?

Scratch projeleri genellikle sahnenin üstündeki **Yeşil Bayrak** simgesine tıklandığında başlar. Bu kod bloğu programın başlamasını tetikler ve sarı renkli **Olaylar** kategorisi altında bulunur.

9. Yeşil Bayrak ve Kırmızı Buton

Yeşil Bayrak: Yazılan kodları baştan çalıştırır (Projeyi Başlatır).
Kırmızı Buton (Sekizgen): Çalışan tüm kodları, döngüleri ve animasyonları anında durdurur (Projeyi Durdurur).

10. Şartlı İşlemler (Eğer... İse)

Kodların belirli bir şarta göre çalışmasını sağlayan karar blokları turuncu renkli **Kontrol** menüsündedir.

11. Operatörler Menüsü (Yeşil)

Matematiksel işlemler ve sayı karşılaştırmaları her zaman yeşil renkli **Operatörler** sekmesinden alınır.

12. Çift İhtimalli Şartlar

Ortada **iki farklı ihtimal** varsa mutlaka **[Eğer ise, Değilse]** bloğu kullanılır.

13. Algılama Menüsü (Değiyor mu?)

Bir kuklanın fare imlecine, ekran kenarına veya başka bir kuklaya dokunup dokunmadığını kontrol eden hissetme blokları açık mavi renkli Algılama menüsündedir.

14. Rastgele Konuma Git

Bu blok çalıştırıldığında kukla yürüyerek gitmez. Sahne içindeki rastgele bir X,Y koordinatına **anında ışınlanır.**

15. Puanı Artırmak (Değiştir Bloğu)

Oyunda elma yedikçe puanın artması için **Değiştir** bloğu kullanılır.

Eğer [Puanı 1 yap] derseniz, ne kadar elma yerseniz yiyin puan hep sabit 1 olarak kalır.

16. Sonsuz Döngü (Sürekli Tekrarla)

İçine yerleştirilen kodların, program durdurulana kadar **baştan sona hiç durmadan** çalışmasını sağlar.

17. Değişken İsimlendirme Sanatı

Programlamada bir değişken oluştururken isimler rastgele verilmemelidir. **İsim, içindeki veriyi anlatmalıdır.** "İki sayının toplamı" programı için şu bloklar oluşturulmalıdır:

18. Hızlı Koordinat Pratiği

Başlangıç **X: 100, Y: 100**

1- 

→ 100'e 50 ekle = 150

2- 

→ Mevcut değeri boşver, direkt -100'e sabitle.

Son Konum: X: 150, Y: -100

19. Değişken ve İşlem Pratiği

Sayı1 = 70, Sayı2 = 30
Ortalama = (Sayı1 + Sayı2) / 2 = 50
Sonuç = Ortalama - Sayı2 = (50 - 30) = 20
Eğer Sonuç > 10 ise Sonuç'u 50 yap: 20 sayısı 10'dan büyük olduğu için sonuç en son **50** olur.

20. Yürüme Animasyonu Nasıl Yapılır?

Bir kuklayı yürütmek için sadece "10 adım git" demek yetmez. Ayaklarını oynatıyormuş gibi bir animasyon oluşturmak için hareket komutunun yanına şu blok eklenmelidir:

sonraki kostüm

21. Uçan Kuş Hatası

Bir kuş kuklası sadece ileriye doğru donuk bir şekilde gidiyorsa, kanat çırpma komutu unutulmuş demektir. Çözüm olarak koda Görünüm menüsünden sonraki kostüm bloğu eklenmelidir.

22. Sahne Dışına Çıkmamak

Kuklanın sahne dışına çıkıp kaybolmasını önlemek için kenara geldiysen sektir komutu kullanılır. Bu komut kukla kenarlara çarptığı an yönünü tersine çevirir.

23. Kenardan Dönünce Tepetaklak Olmak

Kukla kenara çarpıp döndüğünde baş aşağı (tepetaklak) dönebilir. Bunu engellemek ve dik durmasını sağlamak için dönüş stilini sol-sağ yap ayarı kullanılmalıdır.

24. İşlemleri Neden Bekletiriz?

Bilgisayarlar komutları çok hızlı çalıştırır. Eğer art arda çok hızlı kostüm değiştirir veya diyalog kurarsak gözümüz bunu yakalayamaz. Görebilmek için Kontrol menüsünden 1 saniye bekle eklenir.

25. Kuklalar Arası Konuşma (Haber Sal)

İki kuklayı konuştururken "Saniye bekle" yöntemi kullanmak kaymalara sebep olur. Profesyonel iletişim şudur:

KEDİ (Gönderen Kukla):

tıklandığında
merhaba de
haber1 haberini sal

KÖPEK (Alıcı Kukla):

haber1 haberini aldığımda
Selam, hoş geldin! de

26. Birbirini Sıfırlayan Kodlar

Bir kuklaya peş peşe şu komutlar verilirse:

`x konumunu (-10) değiştir`
`x konumunu (10) değiştir`

Sonuç: +10 ve -10 birbirini nötrler ve kukla yerinden hiç kıpırdamamış gibi görünür.

27. 'Adım Git' Bloğunun Yönü

`(10) adım git` komutu her zaman sağa gitmez. Kukla o anki **yöneldiği açı (burnu)** nereyi gösteriyorsa o tarafa doğru ilerler. (0 ise yukarı, -90 ise sola gibi).

28. Açı Bulma Uygulaması (Bölüm 1)

İç İçe "Eğer İse Değilse" yapısı bir şelale gibidir. Yukarıdan kontrol başlar.

```
eğer <Açı < 90> ise
    (haber1) haberini sal → Dar Açı
    değilse
    (Açı 90 veya daha büyükse buraya gir - Devamı aşağıda)
```

Kullanıcı **50** girerse ilk baştaki "Açı $<$ 90 mı?" sorusuna Evet yanıtı alınır. Kukla **Dar Açı** der ve işlem biter, aşağı inmez.

29. Açı Bulma Uygulaması (Bölüm 2)

Açı 90'dan küçük değilse (*değilse kısmı*) içindeki yeni şartlara bakılır:

```
değilse ... eğer <Açı = 90> ise
    (haber2) haberini sal → Dik Açı
    değilse
    (haber3) haberini sal → Geniş Açı
```

Açı **90** girilirse: 90 $<$ 90 değildir. İkinci şarta bakar: 90 = 90 mı? Evet (**Dik Açı**).
Açı **120** girilirse: 120 $<$ 90 değildir, 120 = 90 da değildir. En sondaki *değilse* kısmı çalışır ve kukla **Geniş Açı** der.

30. Metin Kodlamaya Geçiş (Değişkenler)

Bloklardan metin kodlamaya (JavaScript) geçiyoruz! Verileri sakladığımız kutulara **let** kelimesi ile isim veririz.

```
let a = 5;
let b = a + 3;
```

Burada **a** adında bir kutuya 5 koyduk. **b** adındaki kutuya ise a'nın değerinin 3 fazlasını (8) koyduk. Eşittir (=) işareti sağdakini alıp soldakinin içine koyar.

31. Eğer / Değilse (if / else)

Şart bloklarının metin halidir. "Eğer" için **if**, "Değilse" için **else** kullanırız.

```
let a = 0;
if (1 < 5) {
a = 6;
} else {
a = 3;
}
```

Nasıl Çalışır? 1 sayısı 5'ten küçük mü? **EVET.** Şart uyduğu için süslü parantez içine girer ve a'yı 6 yapar.

32. Diziler (Tren Vagonları)

Birden fazla değeri tek bir kutuda sıralamak için köşeli parantez **[]** kullanırız. **Dikkat: Bilgisayarlar saymaya 1'den değil, 0'dan başlar!**

```
let a = [3, 4, 'Z', 8, 7];
a[0] // Sonuç: 3
a[3] // Sonuç: 8
```

33. Uzunluk Ölçme (length)

İngilizcedeki Length kelimesi "Uzunluk" demektir. Bir kelimenin içinde kaç harf olduğunu bulmak için yanına **.length** yazarız.

```
let a = 'nwxrdie';
a.length;
```

Burada harfleri sayarken normal şekilde 1'den başlarız. n(1) w(2) x(3) r(4) d(5) i(6) e(7). Uzunluk = 7 olur.

34. Fonksiyonlar (Fabrikalar)

Fonksiyonlar, içine sayı atıp yeni bir sonuç aldığımız fabrikalardır. **return** komutu, üretilen sonucu bize geri verir.

```
function hello (a, b) {
return a * b;
}
hello(4, 1);
```

Bu fabrikanın görevi içine giren iki malzemeyi (a ve b) çarpmaktır. Sonuç 4 olur.