# İşletim Sistemleri Dönem Ödevi
# 2024-2025 Bahar Dönemi

## C ile Çok Katlı Apartman İnşaası Üzerinden Process , Thread ve Senkronizasyon Kavramlarının Modellenmesi

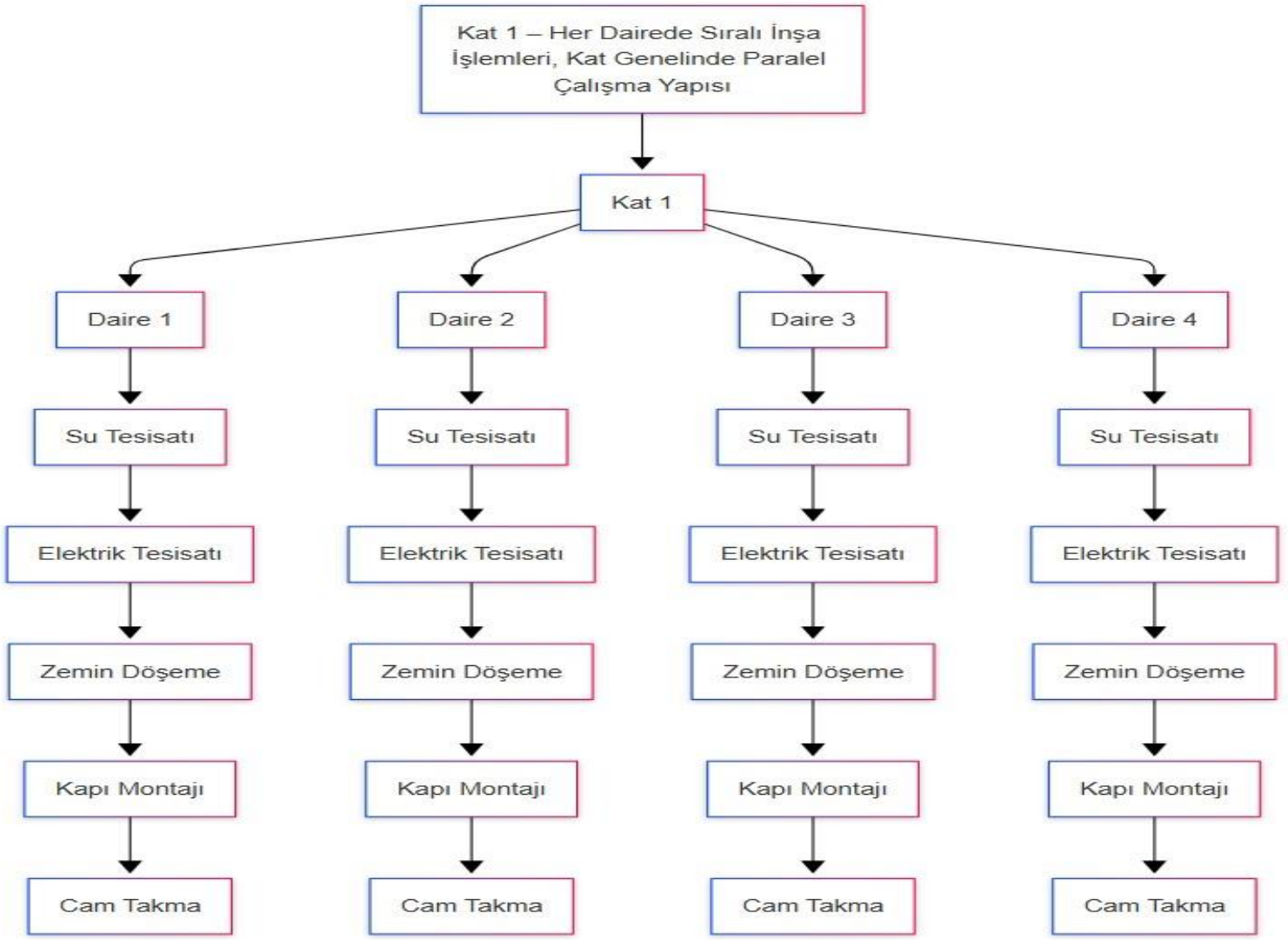
Hasan Umut Kocatepe
21360859077

Halil Alpak
22360859038

# Projenin Amacı

- Bir apartmanın kat ve daire bazlı inşaat süreci, C dili ile paralel programlama yöntemleri kullanılarak modellenmiştir. Process ve thread yapıları bir arada kullanılarak senaryo destekli bir şantiye simülasyonu oluşturulmuştur.

# Kat ve Daire İnşaat Akışı



# Kullanılan Kütüphaneler

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3  #include <unistd.h>
 4  #include <pthread.h>
 5  #include <semaphore.h>
 6  #include <signal.h>
 7  #include <sys/wait.h>
 8  #include <sys/mman.h>
 9  #include <time.h>
10
11  #define KAT_SAYISI 10
12  #define DAIRE_SAYISI 4
```

# KatDurumu Struct'ı ve Mutex , Semafor Tanımları

```
14  typedef struct {
15      int insaat;
16      int boya;
17      int mobilya;
18      int internet;
19  } KatDurumu;
20
21  sem_t sem_vinc;
22  sem_t sem_asansor;
23
24  KatDurumu *kat_durumlari;
25  pthread_mutex_t vinç_mutex, asansör_mutex, boya_mutex, mobilya_mutex, internet_mutex
;
26  sig_atomic_t deprem_var = 0;
27  volatile sig_atomic_t yangin_var = 0;
28  int *kat_bitti;
29  int *boya_bitti;
30
```

# Yangın Kontrol Fonksiyonu

```
31  // Yangin kontrol thread'i: Belirli sure sonra yangin cikar ve digr islemleri gecici
durdurur
32  void *yangin_kontrol(void *arg) {
33      if (rand() % 100 < 70) {  // %70 olasilikla yangin cikmaz
34          pthread_exit(NULL);
35      }
36
37      sleep(rand() % 20 + 5); // Yangin gecikmesi
38
39      int daire = rand() % DAIRE_SAYISI + 1;
40
41      printf("\n[YANGIN] Daire %d'de yangin cikti . Tum isler gecici olarak
durduruluyor!\n", daire);
42      fflush(stdout);
43      yangin_var = 1;
44
45      sleep(0.5); // Müdahale süresi
46      printf("[YANGIN] Yangin sonduruldu, isler devam ediyor.\n");
47      fflush(stdout);
48      yangin_var = 0;
49
50      pthread_exit(NULL);
51  }
52
```

# Deprem Kontrol Fonksiyonu

```
55  void *deprem_kontrol(void *arg) {
56      if (rand() % 100 < 65) {
57          // %65 olasilikla hic deprem olmaz
58          pthread_exit(NULL);
59      }
60
61      sleep(rand() % 15 + 5); // Depremin meydana geldigi zamanisimule et
62
```

```
63      float siddet;
64      int olasilik = rand() % 100;
65
66      if (olasilik < 60) {
67          // %60 olasilikla 3.0-4.9
68          siddet = ((rand() % 20) + 30) / 10.0;
69      } else if (olasilik < 90) {
70          // %30 olasilikla 5.0-6.4
71          siddet = ((rand() % 15) + 50) / 10.0;
72      } else {
73          // %10 olasilikla 6.5-7.5
74          siddet = ((rand() % 11) + 65) / 10.0;
75      }
76
77      printf("\n[DEPREM] %.1f siddetinde deprem meydana geldi!\n", siddet);
78      fflush(stdout);
79      deprem_var = 1;
80
81      if (siddet >= 6.5) {
82          int kayip = rand() % (KAT_SAYISI * DAIRE_SAYISI) + 1;
83          printf("[YIKIM] Bina agir hasar ald? ve coktu! %d kisi hayatini kaybetti.\n"
, kayip);
84          printf("Insaat tamamen iptal edildi. Sistem kapatiliyor...\n");
85          fflush(stdout);
86          kill(0, SIGKILL);
87      } else if (siddet >= 5.0) {
88          printf("[HASAR] Bina orta duzeyde hasar aldi. Guvenlik denetimi
yapilmali.\n");
89
90          sleep(0.5); // guvenlik denetimi icin bekleme
91          printf("[DEPREM] Guvenlik kontrolu tamamlandi. Isler devam ediyor.\n");
92          fflush(stdout);
93          deprem_var = 0;
94      } else {
95          printf("[GUVENL?K] Bina depremi hasarsiz atlatti. Insaata devam ediliyor.\n"
);
96          fflush(stdout);
97          deprem_var = 0;
98      }
99
100     pthread_exit(NULL);
101 }
```

# Daire İşlemleri Threadleri

```c
// Su tesisati islemi: Her dairede baslatilir, yangin varsa beklenir
void *su_tesisati(void *arg) {
    int su_tesisati_daire = *(int *)arg;
    free(arg);
    printf("[Daire %d] Su tesisati basliyor...\n", su_tesisati_daire);
    while (yangin_var || deprem_var) usleep(100000);
    sleep(1);
    printf("[Daire %d] Su tesisati tamamlandi.\n", su_tesisati_daire);
    pthread_exit(NULL);
}

// Elektrik tesisati islemi: Su tesisatindan sonra baslatilir
void *elektrik_tesisati(void *arg) {
    int elektrik_tesisat_daire = *(int *)arg;
    free(arg);
    printf("[Daire %d] Elektrik tesisati basliyor...\n",elektrik_tesisat_daire);
    while (yangin_var || deprem_var) usleep(100000);
    sleep(1);
    printf("[Daire %d] Elektrik tesisati tamamlandi.\n",elektrik_tesisat_daire);
    pthread_exit(NULL);
}
```

```c
// Kap? montaji islemi
void *kapi_montaj(void *arg) {
    int kapi_montaj_daire = *(int *)arg; //
    free(arg); //
    printf("[Daire %d] Kapi montaji basliyor...\n",kapi_montaj_daire);
    while (yangin_var || deprem_var) usleep(100000);
    sleep(1);
    printf("[Daire %d] Kapi montaji tamamlandi.\n",kapi_montaj_daire);
    pthread_exit(NULL);
}

// Cam takma islemi
void *cam_tak(void *arg) {
    int cam_daire_no = *(int *)arg; //
    free(arg);  //
    printf("[Daire %d] Cam takma basliyor...\n", cam_daire_no);
    while (yangin_var || deprem_var) usleep(100000);
    sleep(1);
    printf("[Daire %d] Cam takma tamamlandi.\n",cam_daire_no);
    pthread_exit(NULL);
}
```

```c
// Zemin doseme i?lemi: Su ve elektrik tamamlandiktan sonra yapilir
void *zemin_dose(void *arg) {
    int zemin_dose_daire = *(int *)arg;
    free(arg);
    printf("[Daire %d] Zemin doseme basliyor...\n",zemin_dose_daire);
    while (yangin_var || deprem_var) usleep(100000);
    sleep(1);
    printf("[Daire %d] Zemin doseme tamamlandi.\n",zemin_dose_daire);
    pthread_exit(NULL);
}
```

# daire_insa() Thread'i

```
159 // Daire insa fonksiyonu: 1 dairenin tüm alt islemleri sirasiyla yapilir
160 void *daire_insa(void *arg) {
161     sleep(0.2);
162
163     int daire_no = *(int *)arg;
164     free(arg);
165     printf("[Daire %d] Insa sureci basladi.\n", daire_no);
166
167     pthread_t su_tid, el_tid, zemin_tid, kapi_tid, cam_tid;
168
169     int *su_arg = malloc(sizeof(int));
170     *su_arg = daire_no;
171     pthread_create(&su_tid, NULL, su_tesisati, su_arg);
172     pthread_join(su_tid, NULL);
173
174     int *el_arg = malloc(sizeof(int));
175     *el_arg = daire_no;
176     pthread_create(&el_tid, NULL, elektrik_tesisati, el_arg);
177     pthread_join(el_tid, NULL);
178
179     int *zemin_arg = malloc(sizeof(int));
180     *zemin_arg = daire_no;
181     pthread_create(&zemin_tid, NULL, zemin_dose, zemin_arg);
182     pthread_join(zemin_tid, NULL);
183
184     int *kapi_arg = malloc(sizeof(int));
185     *kapi_arg = daire_no;
186     pthread_create(&kapi_tid, NULL, kapi_montaj, kapi_arg);
187     pthread_join(kapi_tid, NULL);
188
189     int *cam_arg = malloc(sizeof(int));
190     *cam_arg = daire_no;
191     pthread_create(&cam_tid, NULL, cam_tak, cam_arg);
```

```
192     pthread_join(cam_tid, NULL);
193
194     while (yangin_var || deprem_var) usleep(100000);
195     sleep(1);  // 1 saniyelik sivama islem suresi simule edilir
196
197     printf("[Daire %d] Insa sureci tamamlandi.\n", daire_no);
198     pthread_exit(NULL);
199 }
200
```

# Kat Bazlı İşlemler ve Kat Durum Raporu

```c
// Kat boyama islemi: Insaat bitince baslar, mutex ile senkronize
void *boya_kat_thread(void *arg) {
    int kat = *(int *)arg;
    free(arg);
    while (!kat_bitti[kat - 1]) usleep(100000);
    while (yangin_var || deprem_var) usleep(100000);

    pthread_mutex_lock(&boya_mutex);
    printf("[BOYA] Kat %d boyaniyor.\n", kat);
    sleep(1);
    pthread_mutex_unlock(&boya_mutex);

    boya_bitti[kat - 1] = 1;
    kat_durumlari[kat - 1].boya = 1;
    printf("[BOYA] Kat %d boyama tamamlandi.\n", kat);
    pthread_exit(NULL);
}

// Mobilya tasima i?lemi: Her kata paralel olarak yapilir
void *mobilya_tasi(void *arg) {
    int kat = *(int *)arg;
    free(arg);
    while (yangin_var || deprem_var) usleep(100000);

    pthread_mutex_lock(&mobilya_mutex);
    printf("[Mobilya] Kat %d mobilya tasima basladi.\n", kat);
    sleep(1);
    printf("[Mobilya] Kat %d mobilya yerlestirildi.\n", kat);
    pthread_mutex_unlock(&mobilya_mutex);

    kat_durumlari[kat - 1].mobilya = 1;
    pthread_exit(NULL);
}
```

```c
// Internet baglanti islemi: Her kata ayri ayri uygulan?r
void *internet_bagla(void *arg) {
    int kat = *(int *)arg;
    free(arg);
    while (yangin_var || deprem_var) usleep(100000);

    pthread_mutex_lock(&internet_mutex);
    printf("[Internet] Kat %d baglanti kuruluyor...\n", kat);
    sleep(1);
    printf("[Internet] Kat %d baglanti tamamlandi.\n", kat);
    pthread_mutex_unlock(&internet_mutex);

    kat_durumlari[kat - 1].internet = 1;
    pthread_exit(NULL);
}

// Katlarin insaat, boya, mobilya, internet durumlarini yazdirir
void rapor_yazdir() {
    printf("\n[KAT DURUM RAPORU]\n");
    printf("Kat | Insaat | Boya | Mobilya | Internet\n");
    printf("----|--------|------|---------|---------\n");
    for (int i = 0; i < KAT_SAYISI; i++) {
```

```c
        printf(" %2d |   %s   | %s |   %s   |   %s\n",
               i + 1,
               kat_durumlari[i].insaat ? "?" : " ? ",
               kat_durumlari[i].boya ? " ? " : " ? ",
               kat_durumlari[i].mobilya ? " ? " : " ? ",
               kat_durumlari[i].internet ? " ? " : " ? ");
    }
}
```

# kat_insa_et() Fonksiyonu

```
267 // Kat insaat fonksiyonu:
268 // Bu fonksiyon , katin tüm dairelerini insa etmek icin cagrilir.
269 // fork() edilmis process icinde her daireyi paralel thread olarak insa eder
270 void kat_insa_et(int kat_no) {
271     sleep(0.5);
272     srand(time(NULL) + getpid()); // Her alt process'in benzersiz rand() dizisine
sahip olmasi icin
273     printf("\n=== [Kat %d] Insaat Basladi ===\n", kat_no);
274
275     pthread_t daireler[DAIRE_SAYISI];
276
277     for (int i = 0; i < DAIRE_SAYISI; i++) {
278         int *daire_no = malloc(sizeof(int));
279         *daire_no = i + 1;
280         pthread_create(&daireler[i], NULL, daire_insa, daire_no);
281     }
282
283     for (int i = 0; i < DAIRE_SAYISI; i++) {
284         pthread_join(daireler[i], NULL);
285     }
286
287     if (rand() % 100 < 10) {
288         printf("?[KAZA] Kat %d'de isci dustu! 2 saniye duraklama...\n", kat_no);
289         sleep(2);
290     }
291
292     sem_wait(&sem_asansor);
293     printf("[ASANSOR] Kat %d asansor alani kullaniyor...\n", kat_no);
294     sleep(0.5);
295     sem_post(&sem_asansor);
296
297     sem_wait(&sem_vinc);
298     printf("[VINC] Kat %d vinc ile malzeme çekiyor...\n", kat_no);
299     sleep(0.5);
300     sem_post(&sem_vinc);
301
302     kat_bitti[kat_no - 1] = 1;
303     kat_durumlari[kat_no - 1].insaat = 1;
304     printf("=== [Kat %d] Insaat Bitti ===\n", kat_no);
305     exit(0);
306 }
```

# main() - Başlangıç Kısmı

```
308  // Ana fonksiyon: Tum sistemi baslatir, tum islemleri sirasiyla koordine eder
309  int main() {
310      srand(time(NULL));
311
312      // mutex'lerin baslatilmasi
313      pthread_mutex_init(&vinç_mutex, NULL);
314      pthread_mutex_init(&asansör_mutex, NULL);
315      pthread_mutex_init(&boya_mutex, NULL);
316      pthread_mutex_init(&mobilya_mutex, NULL);
317      pthread_mutex_init(&internet_mutex, NULL);
318
319      sem_init(&sem_vinc, 1, 2); // semaforlar?n baslatilmasi
320      sem_init(&sem_asansor, 1, 1);
321
322      // Paylasilan bellekler(Shared Memory) : mmap() ile paylasilan bellekler tahsis
```

```
edilir .
323      // Bu sayede birden fazla process veya thread ayni fiziksel bellek alanina
erisebilir . Veri kopyalamadan senkronize bilgi paylasimi yapilabilir.
324      kat_bitti = mmap(NULL, KAT_SAYISI * sizeof(int), PROT_READ | PROT_WRITE,
325                       MAP_SHARED | MAP_ANONYMOUS, -1, 0);
326      boya_bitti = mmap(NULL, KAT_SAYISI * sizeof(int), PROT_READ | PROT_WRITE,
327                        MAP_SHARED | MAP_ANONYMOUS, -1, 0);
328      kat_durumlari = mmap(NULL, KAT_SAYISI * sizeof(KatDurumu), PROT_READ |
PROT_WRITE,
329                           MAP_SHARED | MAP_ANONYMOUS, -1, 0);
330
```

# main() - İşlem Akışı ve Temizlik Kısımları

```
331     char secim;
332     do {
333         for (int i = 0; i < KAT_SAYISI; i++) { // Tüm katlarin(10 kat) durumlari
sifirlan?r . Onceden insaat yapildiysa temizlenir .
334             kat_bitti[i] = 0;
335             boya_bitti[i] = 0;
336             kat_durumlari[i].insaat = 0;
337             kat_durumlari[i].boya = 0;
338             kat_durumlari[i].mobilya = 0;
339             kat_durumlari[i].internet = 0;
340         }
341
342         printf("\n>>> [BASLANGIC] Apartman insaati basliyor...\n");
343
344         // Paralel olarak deprem ve yangin olasiliklarini yoneten iki ayri thread
baslatilir
345         pthread_t deprem_tid, yangin_tid;
346         pthread_create(&deprem_tid, NULL, deprem_kontrol, NULL);
347         pthread_create(&yangin_tid, NULL, yangin_kontrol, NULL);
348
349         pid_t pids[KAT_SAYISI];
350         for (int i = 1; i <= KAT_SAYISI; i++) {
351             pids[i - 1] = fork(); // Her kat icin fork() ile child process
olusturulur.
352             if (pids[i - 1] == 0) kat_insa_et(i); // kat_insa_et(i) fonksiyonu
sadece cocuk process icinde calisir.
353             else wait(NULL); //Ana process her katin tamamlanmasini sirayla bekler .
Katlar paralel degil , ardisik insa edilir.
354         }
355
356         // Insaati biten her kat icin boyama islemi baslatilir
357         // Boyama islemi her kat icin ayri bir thread ile yapilir.
358         pthread_t boya_threads[KAT_SAYISI];
359         int katlar[KAT_SAYISI];
360         for (int i = 0; i < KAT_SAYISI; i++) {
361             int *kat_no = malloc(sizeof(int));
362             *kat_no = i + 1;
363             pthread_create(&boya_threads[i], NULL, boya_kat_thread, kat_no);
364         }
365         for (int i = 0; i < KAT_SAYISI; i++) {
366             pthread_join(boya_threads[i], NULL); // Tum boyama islemleri tamamlanana
kadar beklenir
367         }
368
369         printf("\n>>> [?SLEM] Boyama tamamlandi. Mobilya ve internet islemleri
baslatiliyor...\n");
370
371         // Her kat icin mobilya ve internet baglantisi islemi icin ayri birer thread
baslatilir
372         pthread_t mobilya_threads[KAT_SAYISI];
373         pthread_t internet_threads[KAT_SAYISI];
374         for (int i = 0; i < KAT_SAYISI; i++) {
375             int *katno1 = malloc(sizeof(int));
376             *katno1 = i + 1;
377             pthread_create(&mobilya_threads[i], NULL, mobilya_tasi, katno1);
```

```
378
379             int *katno2 = malloc(sizeof(int));
380             *katno2 = i + 1;
381             pthread_create(&internet_threads[i], NULL, internet_bagla, katno2);
382         }
383
384         for (int i = 0; i < KAT_SAYISI; i++) { // Tum mobilya tasima ve internet
baglanti islemlerinin bitmesi beklenir
385             pthread_join(mobilya_threads[i], NULL);
386             pthread_join(internet_threads[i], NULL);
387         }
```

```
389         //Katlar islemleri bittikten sonra afet thread'leri iptal edilir.
390         pthread_cancel(deprem_tid);
391         pthread_cancel(yangin_tid);
392
393         printf("\n[TAMAMLANDI] Apartman hazir! Yasanabilir durumda.\n");
394         rapor_yazdir();
395
396         printf("\nApartmani yeniden insa etmek ister misiniz? (E/H): ");
397         scanf(" %c", &secim);
398
399     } while (secim == 'E' || secim == 'e');
400
401     //Tum kaynaklar(mutexler, semaforlar, paylasilan bellekler) serbest birakilir
402     pthread_mutex_destroy(&vinç_mutex);
403     pthread_mutex_destroy(&asansör_mutex);
404     pthread_mutex_destroy(&boya_mutex);
405     pthread_mutex_destroy(&mobilya_mutex);
406     pthread_mutex_destroy(&internet_mutex);
407
408     sem_destroy(&sem_vinc);
409     sem_destroy(&sem_asansor);
410
411     munmap(kat_bitti, KAT_SAYISI * sizeof(int));
412     munmap(boya_bitti, KAT_SAYISI * sizeof(int));
413     munmap(kat_durumlari, KAT_SAYISI * sizeof(KatDurumu));
414
415     printf("\nProgram sonlandirildi. ?yi çalismalar!\n");
416     return 0;
417 }
```

# Terminal Çıktıları - 1
# Genel Akış

```
[Daire 1] Cam takma başlıyor...
[Daire 3] Cam takma başlıyor...
[Daire 1] Cam takma tamamlandı.
[Daire 2] Cam takma tamamlandı.
[Daire 4] Cam takma tamamlandı.
[Daire 3] Cam takma tamamlandı.
[Daire 1] İnşa süreci tamamlandı.
[Daire 4] İnşa süreci tamamlandı.
[Daire 2] İnşa süreci tamamlandı.
[Daire 3] İnşa süreci tamamlandı.
[ASANSÖR] Kat 8 asansör alanı kullanıyor...
[VINÇ] Kat 8 vinç ile malzeme çekiyor...
=== [Kat 8] İnşaat Bitti ===

=== [Kat 9] İnşaat Başladı ===
[Daire 2] İnşa süreci başladı.
[Daire 1] İnşa süreci başladı.
[Daire 2] Elektrik tesisatı başlıyor...
[Daire 3] İnşa süreci başladı.
[Daire 3] Su tesisatı başlıyor...
[Daire 1] Su tesisatı başlıyor...
[Daire 1] Elektrik tesisatı başlıyor...
[Daire 3] Elektrik tesisatı başlıyor...
[Daire 4] İnşa süreci başladı.
[Daire 4] Su tesisatı başlıyor...
[Daire 4] Elektrik tesisatı başlıyor...
[Daire 2] Su tesisatı başlıyor...
[Daire 2] Elektrik tesisatı tamamlandı.
[Daire 3] Su tesisatı tamamlandı.
[Daire 1] Su tesisatı tamamlandı.
[Daire 1] Elektrik tesisatı tamamlandı.
[Daire 1] Zemin döşeme başlıyor...
[Daire 3] Elektrik tesisatı tamamlandı.
[Daire 3] Zemin döşeme başlıyor...
[Daire 4] Su tesisatı tamamlandı.
```

```
[Mobilya] Kat 9 mobilya tasima basladi.
[Internet] Kat 5 baglanti tamamlandı.
[Internet] Kat 9 baglanti kuruluyor...
[Mobilya] Kat 9 mobilya yerlestirildi.
[Mobilya] Kat 10 mobilya tasima basladi.
[Internet] Kat 9 baglanti tamamlandı.
[Internet] Kat 10 baglanti kuruluyor...
[Mobilya] Kat 10 mobilya yerlestirildi.
[Mobilya] Kat 6 mobilya tasima basladi.
[Internet] Kat 10 baglanti tamamlandı.
[Internet] Kat 7 baglanti kuruluyor...
[Mobilya] Kat 6 mobilya yerlestirildi.
[Mobilya] Kat 7 mobilya tasima basladi.
[Internet] Kat 7 baglanti tamamlandı.
[Internet] Kat 8 baglanti kuruluyor...
[Mobilya] Kat 7 mobilya yerlestirildi.
[Internet] Kat 8 baglanti tamamlandı.

[TAMAMLANDI] Apartman hazir! Yasanabilir durumda.

[KAT DURUM RAPORU]
Kat | Insaat | Boya | Mobilya | Internet
----|--------|------|---------|---------
  1 |   ✓    |  ✓   |    ✓    |    ✓
  2 |   ✓    |  ✓   |    ✓    |    ✓
  3 |   ✓    |  ✓   |    ✓    |    ✓
  4 |   ✓    |  ✓   |    ✓    |    ✓
  5 |   ✓    |  ✓   |    ✓    |    ✓
  6 |   ✓    |  ✓   |    ✓    |    ✓
  7 |   ✓    |  ✓   |    ✓    |    ✓
  8 |   ✓    |  ✓   |    ✓    |    ✓
  9 |   ✓    |  ✓   |    ✓    |    ✓
 10 |   ✓    |  ✓   |    ✓    |    ✓

Apartmani yeniden insa etmek ister misiniz? (E/H): H

Program sonlandirildi. Iyi calismalar!
```

# Terminal Çıktıları - 2
# İşçi Kazası ve Yangın Senaryosu

```
[Daire 1] Cam takma tamamlandi.
[Daire 2] Insa sureci tamamlandi.
[Daire 3] Insa sureci tamamlandi.
[Daire 4] Insa sureci tamamlandi.
[Daire 1] Insa sureci tamamlandi.
!![KAZA] Kat 8'de isci dustu! 2 saniye duraklama ...
[ASANSOR] Kat 8 asansor alani kullaniyor...
[VINC] Kat 8 vinc ile malzeme cekiyor...
=== [Kat 8] Insaat Bitti ===
```

```
 [YANGIN] Daire 3'de yangin cikti . Tum isler gecici olarak durduruluyor!
 [YANGIN] Yangin sonduruldu, isler devam ediyor.
 [Daire 4] Su tesisati tamamlandi.
 [Daire 4] Elektrik tesisatı basliyor...
 [Daire 2] Su tesisati tamamlandi.
 [Daire 1] Su tesisati tamamlandi.
 [Daire 3] Su tesisati tamamlandi.
 [Daire 2] Elektrik tesisatı basliyor...
 [Daire 1] Elektrik tesisatı basliyor...
 [Daire 3] Elektrik tesisatı basliyor...
```

# Terminal Çıktıları - 3
# Deprem Senaryoları

1) 5.0 altı deprem

```
[Daire 4] Zemin doseme basliyor...

[DEPREM] 3.4 siddetinde deprem meydana geldi!
[GUVENLIK] Bina depremi hasarsiz atlatti. Insaata devam ediliyor.
[Daire 2] Zemin doseme tamamlandi.
[Daire 2] Kapi montaji basliyor...
```

2) 5.0 ve 6.5 arası deprem

```
[Daire 2] Kapi montaji basliyor...

[DEPREM] 5.8 siddetinde deprem meydana geldi!
[HASAR] Bina orta duzeyde hasar aldi. Guvenlik denetimi yapimali.
[DEPREM] Guvenlik kontrolu tamamlandi. Isler devam ediyor.
[Daire 1] Kapi montaji tamamlandi.
[Daire 1] Cam takma basliyor...
[Daire 3] Kapi montaji tamamlandi.
```

3) 6.5 üstü deprem

```
[Daire 3] Cam takma tamamlandi.
[Daire 4] Cam takma tamamlandi.

[DEPREM] 7.2 siddetinde deprem meydana geldi!
[YIKIM] Bina agir hasar aldi ve coktu! 40 kisi hayatini kaybetti.
Insaat tamamen iptal edildi. Sistem kapatiliyor...
Killed
```

İzlediğiniz için teşekkür ederim