ÖDEV-1

3. Java'nın platform bağımsızlığını nasıl sağladığını anlatınız.(5 PUAN)

Java'da platform bağımsızlığı olmasını şu şekilde açıklayabilirim: Java'da "bir kere yaz her yerde çalışsın" mantığı var. C, C# gibi diller platform bağımlıdır çünkü makinenin işletim sisteminde derlenir. Fakat Java JVM(Java Virtual Machine) üzerinden çalışır. Bu yüzden aynı prgram nerde derlenirse derlensin ay bytecode'u üretir bu yüzden bir kere yazıldıktan sonra istenilen tüm ortamlarda çalıştırmak mümkündür.

4. Java'da heap ve stack kavramlarını örneklerle açıklayın.(5 PUAN)

Eğer program esnasında boyutları bildirilmiş bir değişken kullanıyorsak stack, değişebilir bir değişken kullanıyorsak heap kullanmamız uygundur. Stack kullanılır ve işi bittikten sonra kendini otomatik olarak bellekten yok eder. Fakat heapte bunu yapamayız. Çünkü garbage collector yok.

Stack	Heap
Ram'de tutulur.	Ram'de tutulur.
Oluşturulan değişkenler stack kapsamından çıkınca otomatik olarak yok edilir.	Blok içinde oluşturulan heap değişken bloğun dışına çıkıldığında zaman otomatik olarak yok edilmez.
Heap'e göre oldukça hızlıdır.	Yavaştır.
Stack üzerinde kullanım fazla olduğunda alan yeterli olmayabilirç Mesela 20 boyutlu bir diziye 21 eleman atamak gibi	Doğru kullanılmaması durumunda bellek sorunu yaşatır.
Değişkenler pointer olmadan kullanılabilir.	Değişkenler pointer ile kullanılır.
Derleme zamanında oluşturulur.	Çalışma zamanında oluşturulur.
Kullanacağınız yerin boyutunu tam olarak biliyorsanız Stack'i kullanmak sizin için uygun olacaktır.	İhtiyacınız olan boyutu tam olarak bilmiyorsanız heap kullanımı sizin için uygun.
LIFO mantığında çalışır. (son eklenen ilk çıkar)	

Değer tip(value type) dediğimiz int, short, byte, long, decimal, double , float gibi tipler stackte tutulur.

Referans tipler: tring, array, interface, class, pointer örnektir.

Stack ve heap'in arasındaki en temel farklardan biri heapte veriler karışık bir şekilde saklanırken stackte artan ve azalan adres mantığında (big and little endian) çalışır. Buna bağlı olarak heapte yer alan bir veriye erişmek stackte yer alan bir veriye erişmeye göre daha maliyetlidir.

5. String class'ı nasıl immutable olmayı sağlamaktadır örnek ve çizimlerle açıklayınız.(5 PUAN)

Immutable: bir kere oluşturulduktan sonra durumu(state) değişmeyen sınıf ve yapılardır. Örneğin Integer, Double, Boolean sınıfları ve string sınıfı immutabledır. Bu sınıfların özelliklerini değiştiremeyiz ve bunları kalıtamayız.

6. Java neden çoklu kalıtımı desteklemez açıklayın? (5 PUAN)

Java çoklu kalıtımı (multiple inheritance) desteklemez. Bu kod hatalıdır.

```
public class bilgisayar_mühendisliğ_öğrencisi extends sayısal_öğrenci, üniversite_öğrencisi{

}
```

7. Build Tool nadir? Java ekosistemindeki build toolar neler açıklayın? (5 PUAN)

Java build tool, yazılımı derleme, birleştirme ve dağıtma sürecini otomatikleştiren bir program veya komut satırı yardımcı programıdır.

Build toollar genellikle kaynak koddan bir uygulama ikili dosyası oluşturmak gibi işlemleri otomatikleştirmek için kullanılır.

Derleme araçları yalnızca kod derlemekle sınırlı değildir. Aynı zamanda şu konularda da yardımcı olabilir: paket yönetimi, bağımlılık yönetimi ve sürekli entegrasyon ardışık düzenlerinde.

Sürekli entegrasyon söz konusu olduğunda, derleme araçları, tüm bağımlılıkları ele alarak kodun oluşturulmasında ve paketlenmesinde önemli bir rol oynar.

1.Apaçi Maven

En popüler olanlarından biridir. İlk olarak 2001'de geliştirilmiş ve o zamandan beri Java projeleri oluşturmak için standart kabul edilmiştir. Maven'in ölçeklenebilirliği ve genişletilebilirliği, onu, otomatik yapılara ihtiyaç duyan ancak yazılım mühendisliği ek yüküne ayrılan çok fazla kaynağa sahip olmayan küçük geliştirme ekipleri için çekici bir seçim haline getiriyor.

2.Ant with Ivy

Gradle, ant ve maven kavramları üzerine inşa edilmiştir. Gradle, proje yapılandırmasını bildirmek için Groovy betiklerini kullanır. Gradle, çoklu proje yapıları için tasarlanmıştır ve yapının hangi bölümlerinin güncel olduğunu belirleyerek artımlı yapıları destekler. Ant şu anda çoğunlukla bir miras olarak görülüyor. Gradle oluşturma aracıyla endüstri ilerliyor. Şahsen, Ant ve Maven'in hala kullanabileceğimizi hissediyorum, bu esas olarak projeye bağlı. Bazen Ant ve Gradle, Maven ve Gradle'ın bir kombinasyonunu, hatta üçünü birlikte kullanabiliriz.

3. SBT, Scala tabanlı bir oluşturma aracıdır. En popüler Java Oluşturma Araçları arasında olduğu kabul edilir. Sbt'nin birçok eklentisi vardır ve geliştiricilerin belirli amaçlar için kendi özel görevlerini kolayca oluşturmalarına olanak tanır.

SBT, projeleri Ruby ve JavaScript dahil olmak üzere birden çok JVM dilinde yürütebilir.

SBT'yi bir java projesiyle kullanmanın ana faydalarından biri, bağımlılıkları otomatik olarak indirebilmesi ve bir geliştirme ortamı kurabilmesidir.

Java için SBT, üçüncü taraf geliştiricilerin katkıda bulunduğu birçok eklentiye sahiptir.

4.Ant with Ivy

Ant, derleme dosyasında tanımlanan işlemin yürütülmesine yardımcı olan bir java kitaplığıdır.

Temel olarak Ant, java uygulamaları oluşturmak için kullanılır. Ant çok esnektir; kodlama kuralları veya dizin yapısı gibi herhangi bir kural dayatmaz. Ivy, bir bağımlılık yöneticisi görevi gören Ant'ın bir alt projesidir.

8. Collection framework içerisindeki bütün yapıları önemli methodlarıyla örnekleyip açıklayınız. (20 PUAN)

Collection, Interface olup içinde benzer türden nesneleri belirli şekilde tutacak, nesnelere ait temel davranışları belirler. Collection yapısı Java'da bu ihtiyaçları karşılamak için : add, addAll, remove, clear, isEmpty gibi metotları sunar.

Collection Interface'e ait özellikler;

int size() : Collection içerisinde bulunan elemanların sayısını verir

örnek:

```
List<Integer> list = new ArrayList<Integer>();

list.add(1);
list.add(2);
list.add(3);
int size = list.size();
```

boolean isEmpty() : Collection içerisinde eleman yoksa true değerini return eder.

örnek

```
Collection s1= [];
Collection s2=["örnek"];

System.out.println(s1.isEmpty());
System.out.println(s2.isEmpty());
```

Çıktı:

true

false

boolean contains(Object element) : Parametre olarak verilen eleman Collection içerisinde mevcutsa true değerini dönderir.

```
Collection s1=["örnek"];
```

System.out.println(s1. contains (“örnek”));

System.out.println(s1. contains (“armut”));

Çıktı:

true

false

boolean add(Object element) : Collection yapısına object ekler . Collection’ı implemente eden sınıflar bu metodu implemente etmek zorunda değildir.

Collection s1=[];

s1. add (“örnek”);

Sonuç:
s1 => ["örnek”];

boolean remove(Object element): Collection yapısından object remove eder . Collection ı implemente eden sınıflar bu metodu implemente etmek zorunda değildir.

Collection s1=[“örnek”, “ali”, “veli”];

s1. remove (“örnek”);

Sonuç:
s1 => ["ali”, “veli”];

boolean containsAll(Collections c) : Bulk bir işlemdir . Verilen obje listesinin collection içerisinde var olup olmadığını kontrol eder

Collection s1=[“örnek”, “ali”, “veli”];

s1. containsAll (“örnek”, “ali”);

s1. containsAll (“örnek”, “ömer”);

Sonuç:
True

false

boolean addAll(Collections c) : Parametre olarak verilen listeyi toplu olarak collection içerisine ekler.

Collection s1=[“örnek”];

s1. addAll (“veli”, “ali”);

Sonuç:
s1 => [“örnek”, “veli”, "ali”];

boolean removeAll(Collections c) : Parametre olarak verilen liste içerisindeki elemanları collection içerisinden remove eder.

Collection s1=[“örnek”, “veli”, "ali”];

s1. removeAll (“veli”, “ali”);

Sonuç:
s1 => [“örnek”];

boolean retainAll(Collections c) :Parametre olarak verilen liste ile collection içerisindeki elemanları aynı tutar . Yani parametre olarak verilen listede olmayan elemanlar collection içerisinde çıkartılır .

Collection s1=[“örnek”, “veli”, "ali”];

s1. retainAll (“veli”, “ali”);

Sonuç:
s1 => [“veli”, “ali”];

void clear() : Liste içerisindeki elemanları çıkartır

Collection s1=[“örnek”, “veli”, "ali”];

s1. Clear();

Sonuç:
s1 => [];

Object[] toArray() : List yapısı array yapısına dönüştürülür