3. Java'nın platform bağımsızlığı

Programlamada diller temel itibariyle yüksek seviyeli diller ve düşük seviyeli diller olarak ayırabiliriz. Programlama dilleri konuşma diline daha yakınsa yüksek seviyeli olarak adlandırılır. Düşük seviyeli dil ise makine dilidir ve programlama dillerinin amacı insanlar tarafından daha anlaşılır olan dilleri makine diline çevirmektir.

Orta seviyeli bir programlama dili olan Java yazılan komutları direkt olarak makine diline çevirmek yerine öncelikle bytecode'a çevirir. Bytecode'lar Java Sanal Makineleri (JVM) tarafından makine diline çevrilir. Java gibi ara katman kullanmadan yazılan kodu direkt makine diline çeviren programlama dilleri farklı işletim sistemlerinde farklı sonuçlar verebilmekte veya çalışmamaktadır. JVM ile java bu sorunun önüne geçmiş ve platform bağımsızlığını sağlamıştır.

4. Java'da heap ve stack kavramlarını örneklerle açıklanması

Heap ve stack hafızadır. Ram'da tutulur. Her ikisi de dilimize 'yığın' olarak geçmekteyse de farklılıkları mevcuttur.

- Stack memory kısa süreli hafızadır. İşi bittiğinde silinir ancak heap memory kendiliğinden silinmez.
- Stack memory düzenli bir 'yığındır'. Dik bir kolinin içine tek tek eşya dizmeye benzer. Koliden bir şeye ihtiyaç duyarsanız en son koyduğunuzu çıkartmanız gerekecektir (LİFO) Heap ise kelimenin tam anlamıyla yığındır. Belirli bir düzeni yoktur.
- Stack memory'de int, long gibi primitive data tipleri ve heapte tutulan bilgilerin refereans adresleri bulunurken; Heap memoryde String gibi boyutu belli olmayan non-primitive data tipleri tutulur.
- Stack Memory düzenli olduğu için istediğimiz veriye ulaşmamız kolayken Heap Memoryde dağınık bulunan bilgilere ulaşmak o kadar kolay olmayabilir.
- Stack Memory'nin kapladığı alan bellidir çünkü içinde boyutu belli veri tiplerini bulundurur ancak Heap Memory dinamiktir.

5. String class'ı nasıl immutable olmayı sağladığının örnek ve çizimlerle açıklanması

Dışardan herhangi bir müdahalede bulunulmaması adına değişkenler private olmalı, değişkenlerin setter methodları olmamalı, final olarak işaretlenmeli, ilk değerleri constructur ile verilmeli, classımızı da final işaretlemeliyiz

6. Java'nın neden çoklu kalıtımı desteklemediğinin açıklaması

Java programlama dilinin tercih edilmesi sebeplerinden en önemlileri güvenilir ve basit olmasıdır. Farklı sınıflarda aynı isme sahip metotlar oluşturabiliriz. Bu iki sınıfla çoklu kalıtım yaparak başka bir sınıf türetebilseydik hangi metodun çağırıldığı java tarafından anlaşılmayacak ve karmaşa yaratacaktı. Java, güvenilir ve basit olma özelliklerini zedelememek adına çoklu kalıtımı desteklemez.

7. Build Tool tanımı ve Java ekosistemindeki build toolar

Bir projeyi anlaşılır kılmak, tekrar eden kodlardan kaçınmanın, updateleri tek merkezden(pom.xml) kolayca yapabilmek ve projenin sürdürülebilirliğini sağlamak konusuda Maven gibi Build Tool'lara başvururuz. Bunlar kodlamanın angarya yükünü alan araçlar diyebiliriz. Her proje için yapılması gereken işlemleri otomatik olarak yapar. Büyük projeler için iş yününü azaltırken standarttı sağlamamıza yardımcı olur. Java için Build toollara örnek vermek gerekirse Mavenin yanında Gradle,Ant gibi yardımcılar da mevcuttur.

8. Collection framework içerisindeki bütün yapıları önemli methodlarıyla örneklenmesi

Collections nesnelerden oluşan bir topluluğu bir arada tutan yapılardır. Bu yapıları birkaç ana başlık altında inceleyebiliriz. Bunlardan birincisi SET yapıları, ikincisi ise LİST yapılarıdır. Bir diğer Collecitons yapısı ise QUEUE iken son olarak bahsetmemiz gereken yapı MAP'dir. SET yapılarına örnek olarak HashSet, LinkedHashSet, TreeSet; LİST yapılarına ArrayList, LinkedList yapılarını örnek verebiliriz. MAP yapılarını ise HashMap, TreeMap diye ikiye ayırabiliriz. HashMap yahut HashSet hızlı yapılardır, bu nedenle bize dönen elementler sıralı değillerdir. Sıralı yapıları kullanmamız gerektiği durumlarda TreeSet yahut TreeMap yapılarını kullanabiliriz. İngilizcede 'ağaç' anlamına gelen bu kelime ile başlayan yapılar sıralı döner ancak HashMap veya HashSet yapılarına oranla biraz daha yavaştırlar.

Bazı önemli metotlar da aşağıdaki gibidir;

- Add- koleksiyiyona eklemek için kullanılan metot
- Clear()-Daha önce eklenmiş nesneleri kaldırıır
- Contains- Listede aradığımız nesneyi içerip içermediğini kontrol eder, true veya false döndürür
- Equals-eşitlik için denetler
- isEmpty- Listede nesne olup olmadığını denetler, true-false döndürür
- size-Kolekeksiyonda bulunan öğelerin sayısı
- toArray()-Koleksiyondaki t,m nesneleri içeren bir diziye dönüştürür