1. Pass by value, pass by reference kavramları nedir ? Java' ile ilişkili olarak açıklayınız.

Metodlara gönderilen parametrelerin nasıl aktarılacağı ile ilgilidir.

Pass by value: parametreler geçilmeden önce parametrenin değeri belirlenir. Bu belirlenen değer bellekte bir alana kopyalanır ve daha sonra parametre aktarımı yapılır. Parametre aktarımı yapılırken ise, bu bellek alanının adresinin kendisi değil kopyası metoda gönderilir. Metod için gönderilen parametrenin değeri değiştirilebilir fakat metottan çıktıktan sonra değer değişmeyecek ilk gönderildiği halinde kalacaktır. Java pass by value yöntemini kullanır.

Pass by referance ise tanımlanmış olan değişkenin (instance variable) RAM'de bulunduğu lokasyonu gösteren yapının metoda parametre olarak gönderilmesidir. Dolayısıyla, metodun içerisindeki işlemler değişkenin orjinal değeriyle yapılır ve bunun sonucunda orjinal verinin değeri değişebilir.

2. Immutability nedir, neden önemlidir ? Bir Java sınıfı nasıl immutable yapılır?

Bir objenin değeri değiştirilemiyorsa bu obje immutable objedir.

Bir class'ın immutable olması için şu özellikleri sağlaması gerekir;

- Tüm tanımlanan değişkenler prive olmalıdır.
- Class'ın içinde setter metod tanımlanmamalıdır.
- Class'ın içindeki getter metodları mutable object döndürmemelidirler. Döndürdükleri obje mutlaka immutable olmalıdır.

İmmutable classlar güvenlik sağlarlar. Basit tasarımlarıyla birlikte değişmesini istemediğiniz bir objeyi deişmez olmasını sağlarlar.

3.Framework ve library arasındaki fark nedir?

Framework ve Library 'in ortak noktası, birileri tarafından yazılan ve başkaları tarafından kullanılabilen ortak kod dolarıdır. İkisinde de başkaları tarafından oluşturulmuş kodların kullanımı söz konusudur. Bu kodların çnd classlar veya metodlar mevcut olabilir. Bu noktada ikisi arasında işlevsel anlamda bir fark mevcut değildir. Aslında aynı amaç için kullanılır.

Aralarındaki temel fark ise kütüphanelerin size daha fazla özgürlük vermesidir. Library kullanırken, library size bazı özellikler vererek kullanmanızı sağlar, bu şekilde almış olduğunuzu kodu kendi sisteminize uygularken size kodu nerede ve ne zaman kullanacağınıza karışmaz veya bir diğer deyişle dikte etmez. Framework ise Library'in tersine kullanacağız özelliğe göre kodu nerede ve ne zaman kullanacağınızı söyler, kullanacağız bu işlev, eğer Framework dokümanın belirtildiği gibi kullanılmaz ise kullanım dışı kalır.

4. Java'da Garbage Collector' un görevi nedir ?

Bir Java programı çalıştığında JVM(Java Virtual Machine) , Isletim Sisteminden, memory(hafiza) temin eder. Bu alana Java Heap Memory ya da kısaca heap deriz. Garbage Collector calistiginda amaci Heap'te bulunan ve ulaşılamayan objeleri Heapten silmektir ve bu alani yeni objeler için hazırlamaktır.

5. Memory leak nedir ? Java'da memory leak oluşması mümkün müdür?

"Memory leak", programın kullandığı hafıza ile işi bittiği halde ilgili hafıza bloğunu serbest bırakmaması durumudur. Bu durum;

- Gerçekte kullanılmayan hafıza bloklarının referans edilmeye devam edilmesinden veya
- Native resource'ların düzgün bir şekilde serbest bırakılmamasından

kaynaklanmaktadır. Zaman içerisinde programın sürekli hafıza kullanması, fakat kullandığı hafızayı işi bittiğinde bırakmaması, memory leak'in giderek büyümesine bundan dolayı yeterince uzun süre çalışan uygulamalarda sistem kaynaklarının kritik miktarlarda tüketilmesine ve uygulamanın hata verip kapanmasına sebep olmaktadır.

Java da memory leak oluşması mümkündür. Garbage collector ile bu durum en aza indirilse dahi hali olma ihtimali vardır. Çünkü nesnelere hala başvuruda bulunuluyorsa, bunlar kaldırılmaya uygun değildir. Sonuç olarak, bu kullanılmayan nesneler gereksiz yere bellekte tutulur.

6. Yeni Java sürümleri ne sıklıkla çıkmaktadır ?

Oracle, Long-term support(LTS) yani uzun süre destekli Java sürümlerini her 3 yılda bir çıkaracağını, onun dışında non-LTS sürümleri de her 6 ayda bir çıkaracağını duyurmuş.

(https://tugrulbayrak.medium.com/java-13-ile-gelen-yenilikler-60e0d81f104f)

7. Stack ve Heap nedir ? Java'da hangi yapılar stack ile, hangi yapılar heap ile ilişkilidir?

Yığın (Stack) alanı, işlemci tarafından verilerin geçici olarak saklandığı veya uygulamanın kullandığı değişkenlerin tutulduğu ve büyüklüğü işletim sistemine göre değişen bellek bölgesidir.

Yığının büyüklüğü, yönetimi, programcının değil uygulamanın sorumluluğundadır. Derleyici veya çalıştırma ortamı, program içindeki değişkenleri ve uzunluklarını tarar ve yığın üzerinde ona göre ayırım yaparak yerleştirir. Bu yüzden derleyici, programı oluşturmadan önce yığın üzerinde oluşturulacak verilerin boyut ve ömürlerini bilmek zorundadır. Bir değişkenin hangi bellek bölgesine kaydedileceğini CPU üzerindeki Stack Pointer (SP) denen register belirler. SP, yığın alanının en üst kısmının yani belleğin o anki uygun yerinin adresini gösterir. Depolanacak verilerin eklenmesi veya silinmesiyle SP değeri bir azaltılır veya artırılır. Bütün verilere eğer stack bölgesinde tahsisat yapılmış olsaydı programlarımızın esnekliği azalırdı, zira C# ta bazı nesneler referans yolu ile belirtildikleri için verinin kendisi stack ta bulunmayabilir.

Bu bellek biriminin yığın olarak anılması, LIFO (Son giren ilk çıkar, Last-In First-Out) ilkesine göre çalışmasından kaynaklanmaktadır. CPU tarafından verilerin yığın alanına konulması, Push, alandan alınması, Pop olarak tanımlanır. Bu ekleme ve çıkarma işlemlerinde bellek adresleri değişmez bunun yerine yığın işaretçisi (stack pointer), aşağı yukarı hareket ettirilerek ilgili veriye erişilir veya yeni veri eklenir. Yığın, dinamik değişkenleri saklamanın yanında, yordam çağrıları yaparken, geri dönüş adresini saklamak, yerel değişkenleri depolamak, yordamlara parametre yollamak için de kullanılır. Yığın bellek alanı, program çalışmaya başladığı anda belirlenir ve daha sonra bu alanın boyutu değiştirilemez. Bu alan, işletim sistemi tarafından genellikle kısıtlı şekilde belirlendiği için yığın üzerindeki yoğun ekleme işlemlerinde hafıza birimi taşması sorunu yaşanabilmektedir. Stackoverflow sitesi de tam da bu sorun üzerine kurulmuş bir sitedir. Dolayısıyla yazacağımız programlarda stack bölgesinin yönetiminde ne kadar bilgili olursak, o kadar az sorun yaşarız.

Heap:

Stack bölgelerinde olduğu gibi heap alanlarıda RAM da bulunan hafıza alanlarıdır. Bütün C# nesneleri bu bölgede oluşturulur. Stack ten farklı olarak Heap bölgesinde tahsisatı yapılacak nesnenin derleyici tarafından bilinmesi zorunlu değildir. Bu yüzden, heap bölgesini kullanmak programlarımıza büyük esneklik katmaktadır. Genellikle ne kadar yer kaplayacağı belli olmayan değişkenlerin (nesnelerin) veya büyük verilerin geçici olarak saklanması için kullanılır. Yığın bölgesinin tersine çalıştırma ortamı (JIT), heap alanında ne kadar yer kullanılacağını bilmek zorunda değildir.

New anahtar sözcüğüyle bir nesne oluşturduğumuzda JIT, bu nesne için heap üzerinde yer tahsis eder. İşimiz bitince heap üzerindeki nesnelerimizi sildiğimizde heap biriminin o bölgesini boşaltmış oluruz. Böylece yeni değişkenleri o bölgede saklayabiliriz. Heap alanı, program, bellek üzerinde açık olduğu sürece yaşamaya devam eder. Bu yüzden *genel (global)* ve *statik (static)* değişkenler bu alan içinde tutulur. Heap alanının okunması veya yazması yığın(stack) alanı kadar hızlı değildir. Kısacası, stack, programda tanımlanan sabit uzunluklu değişkenler için tahsis edilmiş bellek birimidir. Heap ise programın kullanımı için tahsis edilmiş genel amaçlı bellek birimidir.

8. OpenJDK ve OracleJDK arasındaki farklar nelerdir?

OpenJDK, açık kaynak kodlu bir JDK'dır diyebiliriz. Açık kaynak kodlu yazlım olduğu için 3. Taraf kişiler de geliştirmeye katkı duyarlar. OpenJDK, JDK'nın üzerinden türetilmiştir. OpenJDK'nın kaynak kodlarını görebilirsiniz ancak JDK nın kaynak kodlarını göremezsiniz. OpenJDK'nın kaynak kodlarında değişiklikler yapabilirsiniz bu değişiklikler sonrasında Java Test Compatibility Kit testinden geçmek gerekiyor.

Oracle JDK, Oracle İkili Kod Lisans Sözleşmesi kapsamında bir ticari lisans gerektirir.Yani açık kaynak kodlu değildir. Kullanımı için belli bir ücret gerektirmektedir. Oracle JDK açık kaynak kodlu değildir, Oracle JDK da Oracle geliştiricilere destek de sağlamaktadır.

9. @FunctionalInterface anotasyonu nerelerde kullanılabilir, neleri sağlar ?

Function interface, içerisinde sadece bir tane abstract metodu olan interface'dir. Eğer ilgili interface'in türetildiği interface'de abstract metot varsa bu durumda da functional interface olur. Functional interface'ler, lambda expression'ların kullanılabilmesi için tanımlanırlar. Functional interface'ler tanımlanırken, @FunctionalInterface anotasyonu kullanması zorunlu değildir. Bu anotasyon sadece validasyon yapma amacıyla kullanılır. Eğer anotasyon eklenirse ve birden fazla abstract metot eklenmeye çalışılırsa, bu durumda compile error verecektir.