Soru 3: Java'nın platform bağımsızlığını nasıl sağladığını açıklayınız.

Cevap: Java Virtual Machine(JVM) olarak bilinen her platform için ayrı ayrı yazılmış olan sanal bir makine sayesinde kodun çalışması için tüm gereklilikler hazır olmuş olur. Bu sayede kodda herhangi bir değişiklik yapmaya gerek kalmadan bir kere yazıp her yerde çalıştırmış oluruz.

Soru 4: Java'da heap ve stack kavramlarını örneklerle açıklayınız.

Cevap: Stack'te primitive tipte veriler tutulurken(byte, short, int, long, double, float gibi "new" keywordü ile bir örneğini oluşturamadığımız veriler) Heap'te referans tipli veriler tutulur(String, List, Map, MyCustomClass gibi "new" keywordü ile bir örneğini oluşturabildiğimiz veriler).

Soru 5: String class'ı nasıl immutable olmayı sağlamaktadır örnek ve çizimlerle açıklayınız.

Cevap: String class'ı içerisinde herhangi bir String'e ekleme yapılmaya çalışıldığında aşağıdaki method çalışmaktadır.

```
public String concat(String str) {
        if (str.isEmpty()) {
            return this;
        }
        return StringConcatHelper.simpleConcat(this, str);
    }
```

Burada eğer ekleme yapılmadıysa yine objenin kendisini döner (değişim yapılmamış olur) ancak ekleme yapıldıysa StringConcatHelper class'ının bir method'u çağırılır.

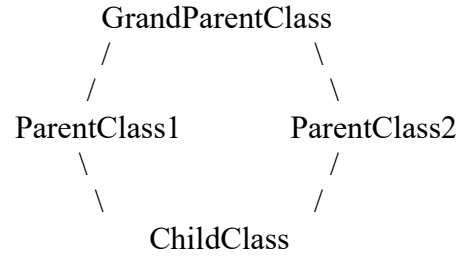
```
  static String simpleConcat(Object first, Object second) {
          String s1 = stringOf(first);
          String s2 = stringOf(second);
          if (s1.isEmpty()) {
              // newly created string required, see JLS 15.18.1
              return new String(s2);
          }
          if (s2.isEmpty()) {
              // newly created string required, see JLS 15.18.1
              return new String(s1);
          }
          // start "mixing" in length and coder or arguments, order is not
          // important
          long indexCoder = mix(initialCoder(), s1);
          indexCoder = mix(indexCoder, s2);
          byte[] buf = newArray(indexCoder);
          // prepend each argument in reverse order, since we prepending
          // from the end of the byte array
          indexCoder = prepend(indexCoder, buf, s2);
          indexCoder = prepend(indexCoder, buf, s1);

          return newString(buf, indexCoder);
}
```

Burada da kırmızı ile yapılan işlemlerden görüldüğü üzere mevcut String objesi üzerinden bir değişim gerçekleşmemekte, her ne olursa olsun [kurallara uygun olduğu sürece (uzunluğu RAM'de tutulabilecek kadar olması gibi)] yeni bir String objesi üretilmektedir. Bu şekilde immutable olmayı sağlamaktadır.

Oluşan String objeleri String pool denilen bir bölgede tutulur. Bu bölgede her objenin yalnızca bir kopyası tutulur. Böylece aynı havuzdan birden fazla değişken için veri saklanabilmiş olur ve hafıza birimi için verimlilik sağlanmış olur. Ayrıca değiştirilemez olması veri güvenliğini de sağlar.

Soru 6: Java neden çoklu kalıtımı desteklemez?

Cevap:



Yukarıdaki şekilde görüldüğü üzere ChildClass hem ParentClass1 hem ParentClass2'den kalıtım almaktadır. Durumu böyle kabul ettiğimizde PC1 ve PC2'de aynı imzayı taşıyan en az bir method olursa ChildClass'tan bu methodu çağırmak istediğimizde hangisinin çağırıldığını anlayamayacaktır. Bu probleme Diamond Problemi denir ve bundan kaçınmak için çoklu kalıtıma izin verilmemiştir.

Soru 7: Build Tool nedir? Java ekosistemindeki build tool'lar nelerdir açıklayın.

Cevap: Build tool'lar uygulamanın derlenmesini otomatize etmek için kullanılan araçlardır. Java ekosisteminde kullanılan build tool'lardan bazıları şunlardır:
- Maven: Bağımlılık yönetim, dokümantasyon ve testing tool'u
- Spring: Temelde Java için bir WebServer oluşturan, veritabanlarına bağlanmayı sağlayan, nesneler ile veritabanında oluşan tabloları birbiri ile eşleştiren bir tool'dur.
- Tomcat: Spring'in default olarak kullandığı bir WebServer'dır.

Soru 8: Collection framework içerisindeki bütün yapıları önemli methodlarıyla örnekleyip açıklayınız.

Cevap: Collection framework içerisindeki önemli yapılar:

Set : Türkçeye "küme" olarak çevirebileceğimiz, içerisinde her elemandan yalnızca bir adet bulunabilen yapılardır. Önemli methodları şunlardır:

addAll(Set anotherSet): içerisine verilen diğer set ile birleşim yapar.
containsAll(Set anotherSet): içerisine verilen seti kapsayıp kapsamadığına bakar.
retainAll(Set anotherSet): içerisine verilen set ile kesişim seti oluşturur (ortak elemanlara bakar)
removeAll(Set anotherSet): içerisine verilen set ile kendisinde bulunan ortak elemanları siler.
add(Object o): Tek bir elemanı eklemek için kullanılır.
remove(Object o): Tek bir elemanı silmek için kullanılır.

Queue: FIFO prensibini benimseyen (First in first out) (İlk eklenen ilk çıkar) yapılardır.

add(Element e): içerisine verilen elemanı ekler, limitli queue'larda IllegalStateException fırlatabilir.
element(): eklenmiş olan ilk elemanı getirir fakat collection'dan silmez. Queue boş ise NoSuchElementException fırlatabilir.
offer(Element e): add() methodu ile aynı işi yapar fakat exception fırlatmaz.
peek(): element() ile aynı işi yapar ancak exception fırlatmaz. Queue boş ise null döner.
remove(): Queue'ya eklenmiş ilk elemanı siler. Eğer queue boş ise NoSuchElementException fırlatabilir.
poll(): Queue'ya eklenmiş ilk elemanı siler ve o elemanı döner. Eğer queue boş ise null döner.

Deque: “Double ended queue” olarak nitelendirebileceğimiz yapılardır. Hem başından hem sonundan ekleme ve çıkarma yapılabilir. Queue'daki methodlara ek olarak şu methodları bulunmaktadır:

contains(Object o): içerisine verilen objenin deque'da olup olmadığını döner.

pop(): Deque'nun ilk indeksinde olan elemanı siler ve döner. Deque boş ise NoSuchElementException fırlatabilir.

push(Object o): Deque'nun ilk indeksine elemanı ekler ve diğer elemanları bu eklenen elemanın arkasına sıralar.

size(): Deque'nun boyutunu döner.

List: Set'lere benzer bir yapı olan List'lerde aynı veriden birden fazla olmasına izin verilmektedir.

toArray(): List'teki tüm elemanları statik boyutlu bir array'e ekleyip bu array'i döner.

subList(int fromIndex, int toIndex): List'in verilen index aralığındaki elemanları döner.

set(int index, E element): List'in belirli bir indeksteki elemanının değerini değiştirmeye yarar.

Map: İçindeki elemanları (Key,Value) ikilisi şeklinde tutan, Set'lerde olduğu gibi birden fazla Key değerine izin vermeyen bir yapıdır. Collection framework içerisinde bulunmamaktadır ancak verileri eşli halde tutabilmek için oldukça kullanışlı bir yapıdır.