Week-1 Yazılı Sorular

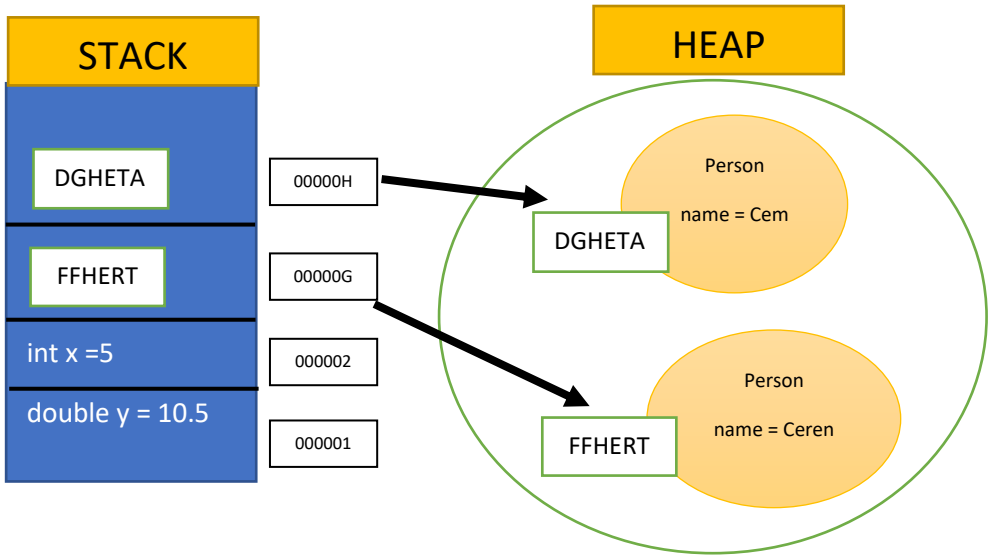
3. Java'nın platform bağımsızlığını nasıl sağladığını anlatınız.

Bilgisayarlar programlama dilini doğrudan anlayamazlar. Dolayısıyla kullanıcılar tarafından yazılan programlama diline ait kodlar bir biçimde makinenin makine anlayacağı hale dönüştürülmelidir. Bilgisayarın anlayacağı sayısal kodlara kodu denir. Java doğrudan makine diline derlenemez. Java kodları .java uzantılı dosyalarda tutulur. Derleyici .java uzantılı dosyaları derler ve bayt kodların bulunduğu .class uzantılı dosyalara çevirir. Java Virtual Machine bilgisayarda bulunan bir sanal makinedir, Java Virtual Machine çalışma anında bayt kodları gerçek makine kodlarına çevirir ve çalıştırır. JVM makineye, işletim sistemine ve işlemciye bağımlılığı ortadan kaldırdığı için yazılan java kodlarını her türlü platformda çalışmaya uygun hale getirir.

4. Java'da heap ve stack kavramlarını örneklerle açıklayın.

Stack ve Heap ram de bulunan mantıksal yapılardır. int, short, byte, long, decimal, double, float gibi tipler value type olarak isimlendirilir ve Stack de tutulur. String, Class type değişkenler referans tiplerdir referansları Stack de değerleri ise Heap de saklanır.

Stack'e erişim Heap'e oranla daha hızlıdır. Stack Last-In-First-Out (LIFO) mantığında çalıştığı için, eklenen son üye Stack'den ilk çıkar. Sırası gelmeden aradaki bir değer ile işlem yapılamaz. Heap'de veriler karışık bir şekilde tutulur. Bu yüzden Heap'deki bir veriye erişmek daha maliyetlidir. Stack'te yer alan veriler direk bellek içine yerleştirilir dolayısıyla erişimi çok hızlıdır. Heap ise çalışma zamanı anında kullanılırlar ve dağınık bir bellek göz yapısı olduğu için erişimi Stack kadar kolay olmaz dolayısıyla yavaş çalışır.



Örneğin yukarıdaki şekli incelediğimizde int tipindeki x ve double tipindeki y değişkeni değer tipinde olduğu için Stack üzerinde yerleştirilmiştir. Class ise referans tipinden bir değişken olduğu için Stack üzerindeki adresi ile Heap'de karşılığı olan referans adresini tutmuştur.

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

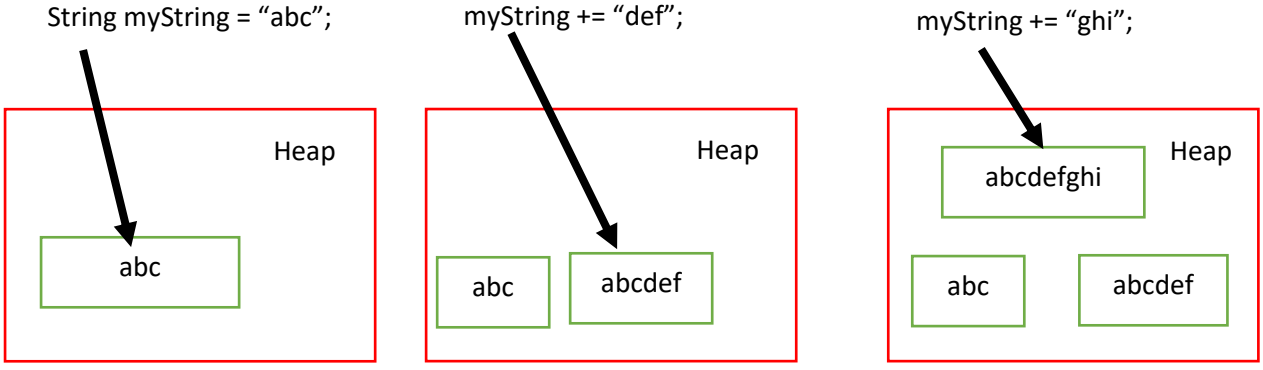
String sınıfı karakter dizilerini temsil eder. "abc" gibi karakter dizinlerini String sınıfının örnekleri olarak uygulanır. String sınıfı sabittir, oluşturulduktan sonra değerleri değiştirilemez. Ancak sadece yaratılmış objenin referansını değiştirebiliriz.

```
The String class represents character strings. All string literals in Java programs, such as "abc", are
implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable
strings. Because String objects are immutable they can be shared. For example:

      String str = "abc";


is equivalent to:

      char data[] = {'a', 'b', 'c'};
      String str = new String(data);
```

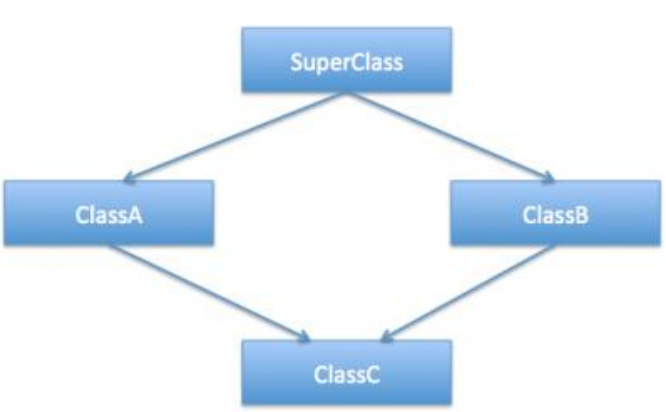
String sınıfı Java'da sabittir. Aşağıdaki resimde String sınıfına bakarsak final olarak tanımlandığını görüyoruz. String'in final yapılmasının nedeni değişmezliği yok etmek ve başkalarının onu genişletmesine izin vermemektir.

```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence,
               Constable, ConstantDesc {

    The value is used for character storage.
```

String nesneleri, String havuzu adı verilen yapıda önbelleğe alınır ve bu String'i sabit hale getirir. Önbelleğe alınmış String değeri birden çok kullanıcıya ulaşır. O yüzden bir kullanıcının yapacağı değişiklik diğer kullanıcıların performansını etkileyecektir. Örneğin bir kişi bir String'in değerini "Kalem" iken onu "KALEM" 'e çevirmiştir. Sistemdeki diğer kullanıcılar da bu değeri okuyacaktır. Yapılan değişiklik sonrası sistemdeki çoğu kullanıcı etkilenmiş oldu. Buradaki örnek çok önemli görünmese de veri tabanı, kullanıcı bilgileri gibi konularda oldukça önemlidir. Bu yüzden String sınıfını "immutable" yapıyoruz. Aşağıdaki şemayı incelersek myString in değeri "abc"dir. myString adlı String'e bir concat (birleştirme) işlemi yapıyoruz ve tuttuğu değerleri sırasıyla değiştiriyoruz. Burada myString'in sadece tuttuğu değerin referansını değiştirmiş olduk. Eski değerler ise belli bir süreliğine Heap'de yer almaktadır.



6. Java neden çoklu kalıtımı desteklemez açıklayın?



Yandaki şekli inceleyelim ve üzerine düşünmeye başlayalım. SuperClass bir abstract (soyut) sınıftır ve soyut print() metodunu içermektedir. A ve B sınıfları (concrete) somut sınıflardır ve SuperClass ı miras almaktadır. SuperClass da bulunan print() metodunu ise override etmektedir.

```
public abstract class SuperClass {

	public abstract void print();

}

public class ClassA extends SuperClass{

	@Override

	public void print(){

		System.out.println("Print Message ClassA");

	}

}

public class ClassB extends SuperClass{

	@Override

	public void print (){

		System.out.println("Print Message ClassB");

	}

}
```

C sınıfının ise hem A hem de B sınıfını miras aldığını varsayalım.

```
public class ClassC extends ClassA, ClassB{

	public void try(){

		print();

	}

}
```

C sınıfında bulunan try() metodu kalıtım yoluyla alınmış print() metodunu tetiklemekte fakat bu print metodunun hangi sınıftan çağrılacağını compiler anlamamaktadır. O yüzden derleme sırasında hata verecektir. Java Karmaşıklığı azaltmak ve dili sadeleştirmek için çoklu kalıtım desteklemez. Javada çoklu kalıtımı arayüzler(interface) sayesinde hallederiz.

7. Build Tool nedir? Java ekosistemindeki build toolar neler açıklayın?

Projemizde yazdığımız kodların derlenmesi sırasında kullanılan otomasyon ve inşa araçlarına build tool denir. Uygulama geliştirirken çeşitli kütüphaneler kullanırız, her kütüphane için gerekli JAR dosyalarını indirmek gerekmektedir ayrıca güncellemeler sonrası sürümlerin de güncellenmesi de gerekecektir. Maven,ANT, Gradle gibi build toollar projeye eklenen bağımlılıklar ile kolay bir şekilde indirmeyi ve projeye yerleştirmeyi sağlar. Bu sayede bizim üzerimizdeki yükü hafifletmiş olur.

İlk build tool 1976 yılında MAKE ismiyle yaratıldı. Daha sonrası ANT,Maven,Ivy ve Gradle ile günümüze kadar geldi.

Another Neat Tool (ANT), java projelerinde kullanılmaktadır. XML tabanlıdır ve Java ile geliştirilmiştir. Taşınabilirlik ve kullanım kolaylığı Ant'in başlıca faydalarından ikisidir. Büyük projelerde build ve deploy işlemlerini büyük oranda kolaylaştırır. Kod derleme, war dosyas hazırlama, test etme oldukça kolaylıklar sağlar.

Maven, POM (Project Object Model)'a dayanan proje yönetim aracıdır. Maven proje oluşturma, bağımlılıklar için kullanılır. ANT gibi iş yükümüzü hafifletir ancak ANT 'dan daha ileri bir teknolojidir. Projeleri kolayca build etmeye, bağımlılıkları yönetmeye, jar, war gibi istediğimiz paket formatında projeyi derlememize yardımcı olur.

POM oluşturduğumuz proje hakkında bilgileri, projenin konfigürasyonu, bağımlılıkları, kullandığımız pluginleri içeren bir xml uzantılı dosyadır. Proje sürümü, açıklama, geliştiriciler, gibi diğer bilgiler de belirtilebilir. Maven bütün işlemleri bu POM.xml dosyasını baz olarak yapar. Maven, geçerli dizinde POM'u arar. POM'u okur, gerekli yapılandırma bilgilerini alır ve ardından hedefi yürütür.

Aşağıda bir pom.xml örneği verilmiştir.

```xml
<project>
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>
</project>
```

Project : projenin root yani kökünü belirtir.

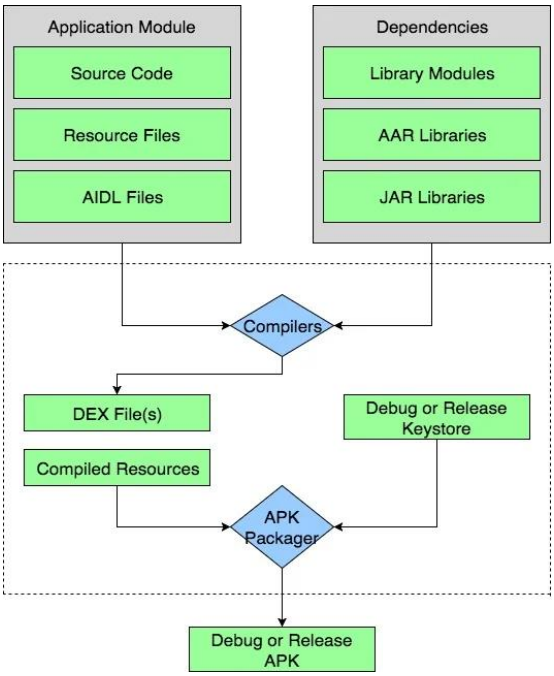
modelVersion - 4.0.0 olarak ayarlaanmalıdır.

groupId - proje grubunun kimliği.

artifactId - yapının kimliği (proje).

version- belirtilen grup altındaki yapının sürümü.

Gradle, Android başta olmak üzere Java, Scala, Groovy, C, C++ gibi birçok dilde kullanılmaktadır. Bir Android projesinin derleme, paket yönetimi ve test etme gibi işlemlerde Gradle bizlere yardımcı olan bir proje inşa sistemidir. Gradle üç ana temel üzerinde çalışmaktadır. Build Anything, Automate Everything ve Deliver Faster. Build Anything (Her şeyi inşa et) istediğimiz dilde çalışacabileceğimizi, istediğimiz platformda paket yönetimi sağlayacağını ifade eder. Automate Everything (Her Şeyi Otomatikleştirin) Gradle zengin API'ya sahiptir, sağladığı eklentiler sayesinde projemizi sistematik hale getirir. Deliver Faster performansı düşürmeden geliştirme safhasını hızlı bir şekilde takip eder.



Kaynak dosyalar, kütüphane dosyaları, proje dosyaları, build dosyaları, referans dosyaları uygulama geliştirmemize olanak sağlıyor. Yukarıda Gradle build process diyagramı gösterilmiştir. Aşağıda ise Gradle projesinin dosya yönetimi gösterilmiştir.

App: Proje modülü.

Build: Build dosyalarını içerir.

Libs: Kütüphane dosyalarını içerir.

Src: Kaynak dosyalarını içerir.

Main: Yazdığımız kodların ve dosyaların bulunduğu klasör.

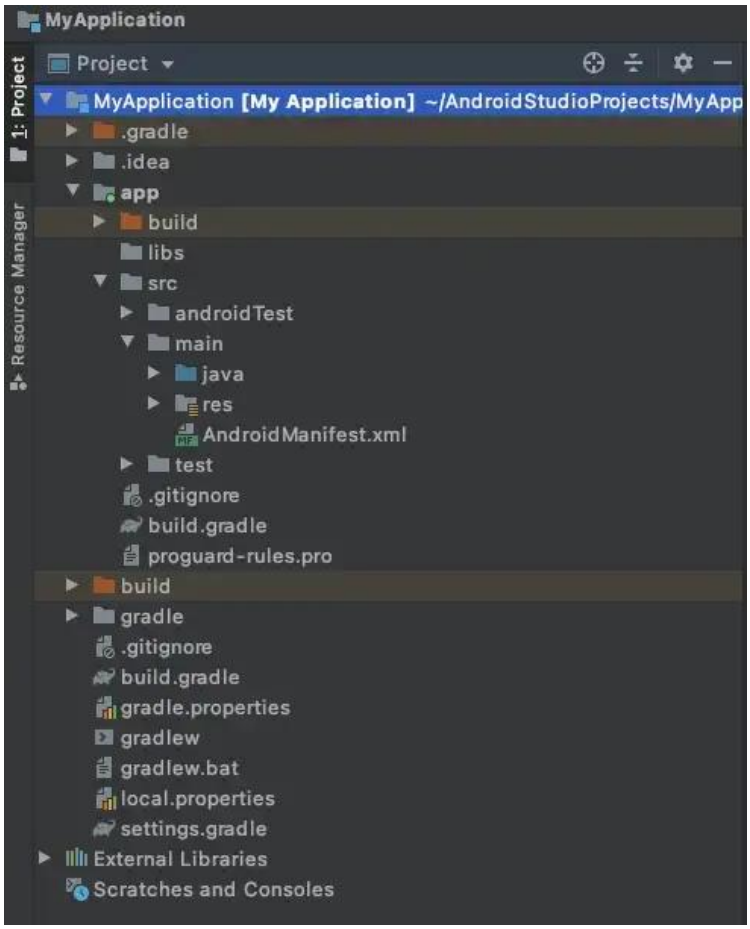
Src/main/java: Yazdığımız kodlar.

Src/main/res: Image,gif gibi dosyaları içerir.

AndroidManifest.xml: Uygulamamızın işlevlerini-gereksinimlerini ifade ederiz.

Test: Test kodlarını içerir.

Build: Uygulamadaki tüm yapılandırmalar buradan yürütülür. Kütüphane eklemeleri, uygulamanın sürümü ve birçok özelliği buradan yapılır.



Kaynaklar:

godoro.com

Yazilimcigenclik.com

Javatpoint.com

Digitalocean.com

maven.apache.org

gradle.org

Medium.com