K147. Kodluyoruz Java Bootcamp
Hafta 1 - Ödev

3) JVM ile java'nın platform bagımsızlıgı saglanır.Java ile yazılan kodu JVM yüklü her makinede çalıştırabiliriz.Kodumuz compiler ile derlendikten sonra bize çıktı olarak byte kodları içeren .class uzantılı dosya verir . Üzerinde çalışılan sistemdeki JVM bu bytecode' u yorumlar ve çalıştırır.
Bu özelik sayesinde Java da "Bir kere yaz, her yerde kullan" özeliğini kullanabiliyoruz.

4) Java'da yeni bir nesne oluşturmak için new anahtar kelimesini kullanırız. Oluşturduğumuz nesneler hafızada tiplerine göre Stack'te veya Heap'te tutulur.

Deger tip nesneler Stack'te tutulur. Java'da Primitive tipler dediğimiz byte, char, int, long, double, boolean gibi tiplere karşılık gelmektedir.

Referans tip nesnelerin değerleri Heap'te referansları ise Stack'te tutulur. Java'da Wrapper tipler Integer, Long, Double, Boolean, Object, Person (bizim tarafımızdan oluşturulan tip) tiplerine karşılık gelmektedir. Örnek :

Memory.java

```java
package com.week1.main;

/**
 * @author hsykrmn
 *
 *  line1 -> main metodu stack bölgesindedir
 *  line2 -> int primitive tip oldugu için stack bölgesinde tutulur
 *  line3 -> new keyword'u kullanıldıgı için heap bölgesinde oluşturulur
 *         ,referansı stack bölgesşnde tutulur.
 *  line4 -> line3 deki durumun aynısıdır.
 *  line5 -> burda verilen parametrenin için yeni referans oluşturulur ve stack'de tutulur
 *  line6 -> foo methodu stack bölgesinde tutulur
 *  line7 -> bu satırdaki string'in referansı stackdedir,kendisi heap bölgesindeki
 *           string pool da tutulur
 */
/*
 * Özet olarak :
 * Stack : Metodlar , metodlara gönderdiğimiz parametreler , local primitive
 * tipler ve local referanslar(object referance) bulunur
 *
 * Heap : Objeler , class variable ve instance variable değerleri tutulur
 */
public class Memory {

    String firstName; // Class 'in variable'ları Heap bölgesindedir.
    int array[] = new int[10];
    static int STATIC_INT = 100;

    public static void main(String[] args) { // Line 1
        int i = 1; // Line 2
        Object obj = new Object(); // Line 3
        Memory mem = new Memory(); // Line 4
        mem.foo(obj); // Line 5
    }

    public void foo(Object param) { // line6
        String str = param.toString(); // Line 7
        System.out.println(str);
    }

}
```

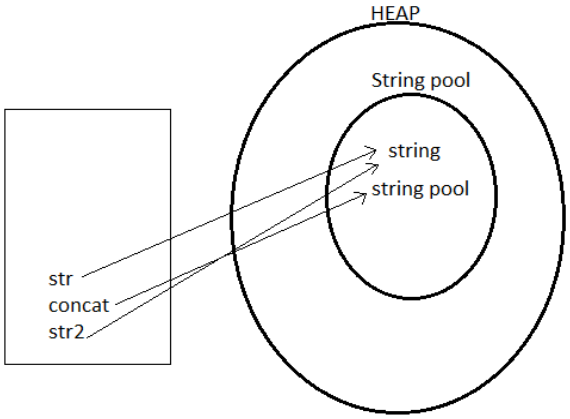
5) Aşagıdaki örnek de str referansına “ pool” kelimesini concat ettik sonuc olarak bize yeni referans dönderiyor önceki atanan deger ve yeni oluşturulan deger Heap bölgesinde bulunan String pool içerisnde yer alır. Eger yeni bir str2 adlı referans oluşturmak istersek ve ona “string” degerini atarsak yine str referansına verilen heap bölgesindeki bölgenin adresi verir . Ve str ile str2 ‘nin adreslerini kıyaslarsak true sonucu gelir.

```
public static void main(String[] args) {
    String str = "string";
    String concat = str.concat(" pool");

    System.out.println(str);
    System.out.println(concat);

    String str2 = "string";

    System.out.println(str == str2);
}
```



6) Belirsizligi ortadan kaldırmak için çoklu kalıtıma izin verilmez örnek verirsek :
class B extends A ,C --> B class’ı Hem A hem C class ‘ini miras alsın ikisinde de display methodu oldugunu varsayalım , eger B class’i display metodunu kullanmak isterse compiler hangi sınıfa ait display methodunu kullanıcagına karar veremez. Böyle durumları ortadan kaldırmak için çoklu kalıtıma izin verilmez

7) Build tool kaynak kod’dan çalıştırılabilir uygulama oluşturma sürecini otomatikleştiren yardımcı programlardı

Yaptıgı temel operasyonlar : Bagımlılıkları indirmek,
Kaynak kodu derleyerek bilgisayarın anlayacagı formata dönüştürmek,
Paketlemek,Testlerin yapılması,
Ürünün ortama deploy edilmesi

Apache Ant, Maven, Gradle yaygın kullanılan java build toollarına örnektir.

Çok kullanılan toollardan biri olan Maven’i incelersek :

Maven kaynak kodlarını alır , derler, ve bize bir cıktı verir. Bu jar,war dosyası olabilmektedir.

Maven bize standart bir dosya dizin yapısı sunmaktadır.

Aşagıdaki resimde bize pom.xml, src klasörü mevcuttur. src klasöründe main klasöründe projemizin kodları yer almaktadır, test kısmında projemiz üzerine yazılan test kodlarını içerir.

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

pom.xml dosyasını incelersek :
içerisinde projemizin ismi, paket ismi
versiyon numarası yer almaktadır ,
Properties bölgesinde kullanılacak olan compiler'in version yer almaktadır,

Dependencies alanı ise projemiz için gerekli bagımlılıkları yönetebilecegimiz alandır.

maven bu özelliklerinin yanı sıra bize Archetypes sunmaktadır. Bu Archetypes hazır şablonlar içermektededir.

emlakcepte/pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.work</groupId>
  <artifactId>emlakcepte</artifactId>
  <version>1.0.0</version>

   <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>6.0.2</version>
        </dependency>

    </dependencies>


</project>
```