1.D

The Java literature frequently refers to the Signature of a method. A method signature is a collection of information about the method, and that includes the name, type (e.g., static or non-static), visibility (e.g., public, private, etc.), arguments (e.g., formal parameters), and return type.

The main method is the entry point of the JVM when the class in launched. The JVM launchs the Java program by invoking the main method of the class identified in the command to start the program. The method main must be declared public, static, and void. It must accept a single argument that is an array of strings.

The main method can be declared as either:

public static void main(String[] args) or public static void main(String args[])

2.A

The diagram don't have anythink with platform independent.

3. C

Yazılan Java kaynak kodları .java uzantılı dosyalarda yer alır ve Java derleyicisi tarafından derlenerek bytecode adı verilen kodları içeren .class uzantılı bir dosya elde edilir.

4.B

When you tried to compile you have error : "The type Date is ambiguous"

5.A

6.D

7.B

java.lang paketi içinde System sınıfı yer almaktadır.. Platformdan bağımsız olarak sistem düzeyindeki eylemleri belirleyen static bir sınıftır.

Bütün java programları java.lang paketinin (importunu otomatik olarak yapar, onu ayrıca import etme gereği yoktur. Dolayısıyla, her java programı, System sınıfını içerir. System sınıfı public damgalı olduğundan, program içindeki her kod ona erişebilir. Ama, ayrıca final damgalı olduğundan, kalıtım olamaz; yani onun alt sınıfları yaratılamaz. Bu nedenle onun metotları da, otomatik olarak, final damgalı olur, yani değiştirilemezler.

System'in üç tane sınıf değişkeni vardır: err, in, out. Bunlardan ilki, sistemde oluşan hataları bildirir, ikincisi giriş akımlarını, üçüncüsü çıkış akımlarını yapar.

8.C

"#" isn't a valid for comment in Java.

9.D

Java dosyaları sadece 1 public class içerebilirler ve dosya asının class adıyla aynı olması zorunludur.

10.B

```
public class Q10 {
static String weight = "A lot"; // is static and class variable.
/* default */ double ageMonths = 5, ageDays = 2; // are instance variable.
private static boolean success = true; // is static and class variable.
public void main(String[] args) {
final String retries = "1"; // is local variable.
// P1   }}
```

11.B
Kullanılmayan importların silinmesinin uygulamayı derlemeye bir etkisi olmamaktadır.
12.A
statik sınıfın bir örneği olmadan statik olmayan bir değişkene erişmeye çalıştığından kod derlenmez.
13.D
Kodların çalıştırılması için derlenmesi gerekmektedir. Derlendikten osnra da uzantıları .class olmaktadır. Bu yüzden 1 yanlış.
Java nesne yönelimli olduğu için 2 yanlış
Kodların derlenmesi için JVM ihtiyacı vardır. 3 yanlış.
14.D
Değişken tanımlamaları class içerisinde yapılmalıdır.
15.C
Bir paket içerisine farklı paketler import edilebilir erişim belirteçleriyle de erişim yetkileri kontrol edilebilir.
16.B
Komut satırından bulunduğun dizindeki “Manager.java” kodunu derlemek için “javac Manager.java” yazmak gerekir. Eğer bir hatayla karşılaşmazsan “java Manager” komutyla da kodu çalıştırabilirsin.
17.D
Java Encapsulation java sınıfı içerisinde tanımladığımız değişkenlerimizi korumak veya saklamak anlamanına gelmektedir. Tanımladığımız bir sınıf içerisindeki değişkenin direk olarak değiştirilememesi, bunun yerine bizim izin verdiğimiz ölçüde, metotlar aracılığıyla değiştirilmesidir.
18.D
“height” değişkeni if döngüsü içinde anımlandığı için dıarıdan erişelemyecek ve hata verecektir.
19.A
.java uzantılı dosyalar java da derlendıkten sonra .class uzantılı bytecode dosyalarına donüşürler. Bu dosyalar da JVM olan herhangi bir bilgisayarda çalışırılabilir.
20.D
Javada satır sonlarına nktalı virgül (;) konularak satır ifade bitirilir.
21.C
today =20, tools.tomorrow =10, tools.yesterday=1
Kod calıstırıldıgında sonuc 31 cıkmaktadır.
22.C
1.satır class ifadesi eksik
2.satır hem double hem de int kullanılış
4.satır private yanlıs yerde kullanılmıs.
23.D
Platform bağımsızlığı özelliği sayesinde java uygulamalarını JVM olan bilgisayarlarda

çalıştırabilirz.

24.A

25.B

İnstance ve local veriable belirli yerlerde kullanılırken class veriable ları heryerde kullanılabilmektedir.

26.?

27.D

Java source files have the following ordering:
Beginning comments
Package and Import statements
Class and interface declarations

28.D

Kodunuzda import java.lang eklemesi yapmanıza gerek yoktur. Çünkü kodunuz derlenirken java.lang paketinin altındaki sınıflar otomatik olarak JVM tarafından kodunuza eklenir çünkü java.lang, Java uygulamalarının çalışması için gerekli en temel pakettir. Java'daki tüm sınıfların atası Object sınıfı dahi, java.lang paketinin içindedir.
İmport.stars.* ile de stars icindekileri almis oluruz. Ayrica import.stars.Blackhole yazmamiza gerek yoktur.

29.?

30.B

The javac command reads source files that contain module, package and type declarations written in the Java programming language, and compiles them into class files that run on the Java Virtual Machine.

31.B

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

32.D

```
package agent;  //en basta package olmalı
public class Banker {  // class ismi kullanılmalı
        private static long getMaxWithdrawal() {
        return 10;
}}
```

33.A

```
package HW1;
public class Q33 {
        static int start = 2;
        final int end;

        public Q33(int x) {
                x = 4;
                end = x;
        }
        public void fly(int distance) {
                System.out.print(end - start + "");
                System.out.print(distance);
        }

        public static void main(String... start) {
                new Q33(10).fly(5);

        }
```

```
}
```

34.D

Programcı, yeni alt-sınıfları tanımlarken, üst-sınıftan (superclass) kalıtsal olarak geleceklere ek olarak, kendisine gerekli olan başka değişken ve metotları da tanımlayabilir.
Bu yolla, bir kez kurulmuş olan sınıfın tekrar tekrar kullanılması olanaklı olur. Böylece, programlar daha kısa olur, programın yazılma zamanı azalır ve gerektiğinde değiştirilmesi ve onarılması (debug) kolay olur.

35.A

Yorum satırı yaparken en az 2 tane olacak sekilde istediğimiz kadar "/" ıfadesınden kullanabiliriz.

36.B

public static void main(String arguments) kullanımı yanlıstır. String[] seklınde olmalıydı.

37.B

```
public class Q37 {
        // line a2
        public String color;
        public void attach() {
        // line a3
        }
        // line a4
        public String color;
}

public class Q37 {
        // line a2
        //public String color;
        public void attach() {
        // line a3
        }
        // line a4
        public String color;
}
```

38.A

```
class A {}  //class yandaki gibi  oluşturulabilir. package statement,import statement ya da public modifier zorunlu değildir.
```

39.D

Yazılan Java kaynak kodları .java uzantılı dosyalarda yer alır ve Java derleyicisi tarafından derlenerek bytecode adı verilen kodları içeren .class uzantılı bir dosya elde edilir.

40.A

```
package pocket;
import pocket.complex.*;  ///pocket.complex tanımlı olmadığı için haa verir.
import java.util.*;
public class Q40 {
        public static void main(String[] args) {
                System.out.print(Math.floor(5));
        }
}
```

41.A

42.B

Nesne Yönelimli Programlama (NYP) mantıksal işlemlerden ziyade, nesnelere (object) ve nesneler üzerinde işlemlere odaklanan programlama dili modelidir. NYP'de programlar, nesnelerin birbirileriyle etkileşime geçmeleri sağlanmasıyla tasarlanır. NYP teorisinde 4 temel özelliğin gerçekleştirilmesi zorunlu sayılmıştır ve biri bile eksik ise bu dil saf NYP sayılmamıştır. Bunlar: **Encapsulation, Inheritance, Polymorphism, Abstraction**'dır.

Encapsulation (sarma) NYP'nin temel kavramlarından biridir. Genel tanımıyla kullanıcı tarafından verilerin, sınıfların ve metotların ne kadarının görüntülenebileceği ve değiştirilebileceğinin sınırlarının konulmasını sağlar. Public (herkese açık), private (özel) ve

protected (koruma altında) olmak üzere üç adet access modifier'dan (erişim dönüştürücüsü) bahsedilebilir.
Inheritance (kalıtım) bir sınıftan başka bir sınıf türetirken aralarında bir alt-üst ilişkisi oluşturmayı ve bu sınıflar üzerinde ortak metotlar ve özellikler kullanılmasını sağlayan bir mekanizmadır. NYP'nin temel kavramlarından biridir. Hali hazırda var olan sınıfların üzerine başka sınıfların inşa edilmesini sağlar. 5 çeşit Inheritance çeşidinden söz edilebilir.
• **Single Inheritance (Tekli Kalıtım):** Alt sınıf tek bir üst sınıfın tüm özelliklerini taşır.
• **Multiple Inheritance (Çoklu Kalıtım):** Bir alt sınıf birden fazla üst sınıfın tüm özelliklerini taşır.
• **Multilevel Inheritance (Çok Seviyeli Kalıtım):** Bir sınıfın alt sınıfı oluşturulduktan sonra bu alt sınfın da bir alt sınıfının oluşturulmasına denir.
• **Hierarchical Inheritance (Hiyerarşik Kalıtım):** Bir üst sınıfın birden fazla alt sınıfa base class (temel sınıf)'lik yapmasına denir.
• **Hybrid Inheritance (Melez Kalıtım):** Öbür Inheritance türlerinin 2 veya daha fazlasını barındıran Inheritance türüdür.

Polymorphism (çok biçimlilik) NYP'de programlama dilinin farklı tip verileri ve sınıfları farklı şekilde işleme yeteneğini belirten özelliğidir. Daha belirgin olmak gerekirse, **metotları** ve **türetilmiş sınıfları** yeniden tanımlama yeteneğidir. Örnek olarak şekil diye bir sınıf olsun; **polymorphism** sayesinde programcı farklı şekillerin alanlarını **farklı metotlar** ile belirleyebilir. Şeklin ne olduğu fark etmeksizin program kullanıcıya doğru alanı verecektir.

Abstraction (soyutlama) NYP'nin temel kavramlarından biridir. **Alt sınıfların** ortak özelliklerini ve işlevlerini taşıyan ancak henüz bir **nesnesi** olmayan bir **üst sınıf** oluşturmak istenirse bir **soyut (abstract) üst sınıf** oluşturulur. **Soyut sınıfın yöntemleri alt sınıfları** tarafından üzerine yazılmak üzere şablon olarak tanımlanabilir veya **soyut metot** olarak oluşturulabilir. **Soyut metota** sahip bir sınıf otomatik olarak kendisi de **soyut** hale gelir ve **soyut sınıflardan nesne** oluşturulmaz.

43.A

```
import food.vegetables.*;  // Broccoli b için gerekli
import food.fruit.*;       // Apple a için gerekli
import java.util.Date;   // Date c için gerekli
```

44.C

```
package HW1;
public class Q44 {
private boolean numLock = true;
static boolean capLock = false;
public static void main(String... shortcuts) {
System.out.print(numLock+""+capLock);
//numLock degiskeni static olmadigi icin static olan bir method da calisamaz.
//"Cannot make a static reference to the non-static field numLock"
}
}
```

45.D

```
package HW1;
public class Q45 {
        static int wheels = 1;
        int tracks = 5;
        public static void main(String[] arguments) {
                Q45 s = new Q45();
                int feet = 4, tracks = 15;
                System.out.print(feet + tracks + s.wheels);
   //4+15+1 =20
        }
}
```

46.B

```
package HW1;
public class Q46 {
```

```
String color = "red";
private void printColor(String color) {
color = "purple"; // gelen blue purple olarak değiştiriliyor ve yazdırılıyr
System.out.print(color);
}
public static void main(String[] rider) {
new Q46().printColor("blue");    //printColor methoduna blue rengini gönderiyor
}
}
```

47.C

Kod calıstırılırken paketeri ayırmak için nokta(.) kullanılır
Kod derlenirken .java dosyaları .class dosyalarına donüşür.

48.B

```
public class Q48 {
        private static boolean heatWave = true;

        public static void main() {
                boolean heatWave = false; //heatWave false yapıldıgı için false yazdırılacak
                System.out.print(heatWave);
        }
}
```

49.C

Book class ı numberOfPages isimli public (+) bir değişkene ve geRating isimli public (+) methoda sahip olmalıdır.
(+) public
(-) private
(#)protected
(~)package/default

50.?