1.A

double ve int farklı tipte değişkenlerdir ve farklı tüde değişkenler aynı satırda tanımlanamazlar double dan sonra da noktalı virgül(;) ile bitirilmesi gerekirdi.

2.D

String table = "metal" olarak bir başlangıc degeri verilmiştir ama String chair baslangıc degerine sahip değildir. Bu yüzden kod derlenmeyecektir.

3.B

Class'ın bir field'i (alanı) olarak tanımlanan değişkenlere **instance variable** denir. Bu alanların değerlerini tanımlarken, constructor'da veya daha sonra herhangi bir metodun içerisinde verilebilir. Hatta herhangi bir değer atanmazsa ilgili veri tipinin varsayılan değeri atanacaktır. Örneğin: int için 0, double için 0.0, floa için 0.0 ve string için null değerleri default değerlerdir.

Fakat final olan bir instance variable tanımlanmışsa bu değer ya tanımlanırken yada constructor'da atanmalıdır. Eğer bir final instance variable tanımlanırken değeri atanmamışsa, tüm constructor'larda ilgili değişken parametre geçilerek değeri atanmalıdır.

4.B

Değişken isimleri rakamlarla başlayamaz.

5.B

Proje İsimleri :

– Küçük ya da büyük harfle başlayabilir.

– Genellikle büyük harfle başlatılır.

– Eğer birden fazla kelimeden oluşuyorsa sonraki kelimelerin baş harfleri büyültülür.

– Geri kalan kısmı küçük yazılır.

Paket İsimleri :

– Küçük harfle başlar.

– Birden fazla kelimeden oluşuyorsa diğer kelimelerin sadece baş harfleri büyültülür.

– Paket isimlendirmeleri yapılırken domain ismi tersten yazılır.

Örnek : tr.com.infopark.business.* gibi

Sınıf isimleri :

– Büyük harfle başlar küçük harfle devam eder.

– Eğer birden fazla kelimeden oluşuyorsa diğer kelimelerin baş harfleri büyültülür.

– Sınıf isminin dosya ismi ile aynı olması gerekir.

Metot isimleri :

– Küçük harfle başlar.

– Birden fazla kelimeden oluşuyorsa diğer kelimelerin sadece baş harfleri büyültülür.

Değişken isimleri :

– Küçük harfle başlar.

– Birden fazla kelimeden oluşuyorsa diğer kelimelerin sadece baş harfleri büyültülür.

– $ ve _ dışında değişken ismi başında, başka herhangi bir karakter kullanılamaz.

– Değişken ismi içinde sayısal bir ifade kullanılacaksa değişken isminin başında kullanılamaz. Ortasında ya da sonunda sayısal ifade geçebilir.

6.B

```
public class HW2Q6 {
	public String convert(int value) {
		return value.toString();
		// int veri tipinistringe çevirirken String.valueOf(value) şeklinde kullanılmalıydı
	}

	public String convert(Integer value) {
		return value.toString();
	}

	public String convert(Object value) {
		return value.toString();
	}
}
```

7.C

Sayısal değerlerde basamaklar arasında alçizgi (_) konulabilmektedir. 9_9_9 değerini çalıştırdığımızda 999 değeriini elde ederiz. Ama altçizgi (_) ifadesiyle sayısal değeri başlatamayız.

8.C

“int” yerine “Integer” , double yerine “Double” gibi büyük harfle kullanılan ifadeler primitive veri tiplerinin Wrapper (Kapsayıcı) sınıfları olarak ifade edilir ve referans tipindedidir. Bu sınıfları kullanarak Max değer Min değer ve Boyut bilgisini gibi değerler bulunabilir.

9.C

```
public class Q9 {
	integer a = Integer.valueOf("1");
	//integer tanımlı bir isim değildir.
	// integer yazan yerler int ile değişirilirse kod derlenmekte ve sonuc 5 çıkmaktadır.

	public static void main(String[] nums) {
```

```
                integer a = Integer.valueOf("2");
                integer b = Integer.valueOf("3");
                System.out.println(a + b);
        }

}
```

10.B

new anahtar sözcüğüyle yeni bir primitive veri tipi oluştururuz.

11.D

float veri tipinin sonuna “f” veya “F” koyulmadığında Java bu tanımlamadaki veri tipini double olarak algılayacağı için geliştirme anında hata verecektir.

dogrusu float f2 = 5.0f şeklkinde olmalıydı.

12.A

byte 8 bit

char 16 bit

float 32 bit

int 32 bit

double 64 bit

13.D

Metotlar belirli vazifeleri verdiğimiz, işlemler sonucunda bize bir geri dönüş değeri verebilen ya da geri dönüşü olmadan belirli işlemleri tamamlayan yapılardı. Programımızda ihtiyaç olduğu yerde metotları çağırarak işlemleri gerçekleştiriyoruz. Fakat bazen bizim metodu çağırmamıza gerek kalmadan, bir sınıftan nesne oluşturduğumuz anda bazı işlemlerin yerine getirilmiş halde olmasını isteyebiliriz. Bu durumlarda kullandığımız metot constructor (yapılandırıcı) metotlardır. Bir yapılandırıcının yaptığı iş, bir nesneyi ilk kullanıma hazırlamaktır.

Yapılandırıcı metotları şu şekilde özelenebilir:

· Yapılandırıcıların erişim belirteci mutlaka ama mutlaka public olmalıdır.

· Yapılandırıcıların adı sınıfın adıyla aynı olmalıdır.

· Yapılandırıcı metot çağrılırken new anahtar sözcüğü kullanılır.

· Yapılandırıcılar bellekte nesneye bir yer ayrılmasını sağlarlar.

· Yapılandırıcılar her çağrılışlarında yeni bir nesne oluştururlar.

Bu soruda sorulan constructor, instance variable ve method names arasında nasıl bir sıralama olması gerektiğidir. Ama bunlar arasında bir öncelik sıralamsı söz konusu değildir. İsenilen yerde kulanılablirler.

14.B

tek satırda iki farklı tür tanımlanmaya calışılmış, ilk tanımlamadan sonra da ; ile bitirilmesi gerekirdi.

15.C

İnstance : ingilizcesi örnek demektir, nesne tabanlı (oob) programlamada bir nesnenin instance ını oluşturmak bir örneğini yaratmak demektir. Mesela; bir masa(table) objemiz olsun. Her instance ını yarattığımızda bu objenin birebir aynı özelliğe sahip bir çok örnekleri oluşturmuş olursunuz. Ürettiğiniz örnek objenin üzerinde istediğinizi yapabilirsiniz. Masayı bir yemek masası, toplantı masası, çoçuk masası vs. özelliği katabilmek için özelliklerini daha sonrasında setleyebilirsiniz. Basitçe;

Masa masa = new Masa(); Masa objesinin bir örneğini oluşturduk.

İki tür initialization block vardır;

* "**instance initialization block**" obje her instantiate edildiğinde çalısır.

*"**static initialization block**" class initialize edildiğinde bir kere çalışır

Ve bunlar süslü parantezle ({}) başlatılırlar.

```
public class Q15 {

	{		//illustrate instance initializers
		System.out.println();
	}

	public Q15() {		//constructor
		System.out.println();
	}

	static {	//static initializers
		System.out.println();
	}
	{		//illustrate instance initializers
		System.out.println();
	}
}
```

16.D

Boşluğa double, short ve int değişkenlerinden herhangi birini koyunca sonuç sıfır çıkmaktadır.

17.A

Java dili kullanılmayan nesneleri silmek için arka planda çalışan bir sisteme sahiptir. Bu yüzden C++ gibi dillerde dinamik nesne tanımlarında mutlaka yer alması gereken nesneyi

silme metodları javada o kadar gerekli değildir. Fakat eğer kendiniz mutlaka bir nesneyi acil olarak silmek istesek finalize() isimli bir metodu sınıfınızda tanımlayıp kullanabiliriz.

Bu metot en fazla bir kere çağrılabilir.

18.D

String normal bir class tır. Diğeleri wrapper class tır.

19.???????????

20.C

float kullanmak için pi = 3.14f; şeklinde tanımlanmalıydı.

21.B

```
public class Q46 {
        public static void main(String[] args) {
                int Integer = 0; // k1
                Integer int = 0; // k2
                //int ismi java da tanımlı bir isim olduğu için bu şekilde kullanılamaz
                Integer ++; // k3
                int++; // k4
                }

}
```

22.???

23.C

5.sorudaki kurallar nurada da geçerlidir.

$ ve _ dışında class ismi başında, başka herhangi bir karakter kullanılamaz
class ismi içinde sayısal bir ifade kullanılacaksa class isminin başında kullanılamaz. Ortasında ya da sonunda sayısal ifade geçebilir.

24.C

25.C

Yerel değişkenlerin default değerleri olmaz.

Alttaki açıklamalar https://www.tutorialspoint.com/java/java_variable_types.htm sitesinden alınmıştır.

There are three kinds of variables in Java

- Local variables
- Instance variables
- Class/Static variables

Local Variables

- Local variables are declared in methods, constructors, or blocks.
- Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block.
- Access modifiers cannot be used for local variables.
- Local variables are visible only within the declared method, constructor, or block.
- Local variables are implemented at stack level internally.
- **There is no default value for local variables**, so local variables should be declared and an initial value should be assigned before the first use.

Instance Variables

- Instance variables are declared in a class, but outside a method, constructor or any block.
- When a space is allocated for an object in the heap, a slot for each instance variable value is created.
- Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
- Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
- **Instance variables can be declared in class level before or after use**.
- Access modifiers can be given for instance variables.
- The instance variables are visible for all methods, constructors and block in the class. Normally, it is recommended to make these variables private (access level). However, visibility for subclasses can be given for these variables with the use of access modifiers.
- Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object references it is null. Values can be assigned during the declaration or within the constructor.
- Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. *ObjectReference.VariableName*.

Class/Static Variables

- Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
- There would only be one copy of each class variable per class, regardless of how many objects are created from it.

- Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.
- Static variables are stored in the static memory. It is rare to use static variables other than declared final and used as either public or private constants.
- Static variables are created when the program starts and destroyed when the program stops.
- Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
- **Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null.** Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name *ClassName.VariableName*.
- When declaring class variables as public static final, then variable names (constants) are all in upper case. If the static variables are not public and final, the naming syntax is the same as instance and local variables.

26. D

25.soruda verilen bilgilere göre static variables ların default değerleri de instance variable lar gibidir. Yani sayısal değerler 0'dır.

27.C

Java da veri tipleri 3 e ayrılır. Bunlar;

*İlkel(Primitif) veri tipleri

*Referans tipler

*Null Veri tipi

Java gibi nesneye yönelimli programlama dillerinde veri tiplerinin hepsi bir sınıftır ancak çok sık kullanıldıkları için Java bazı veri tiplerine ayrıcalık tanır bu türden veri tiplerine temel(primitif) veri tipleri denir. Java dili bu veri tiplerine onlara ait nesneler yaratılmaksızın kullanılmasına olanak sağlar. 8 tane ilkel veri tipi vardır. (byte, short, int, double, float, double, char, boolean)

Referans Veri Tipleri

Temelde 3 tane referans tip vardır. (Class, Array, Interface) Bu 3 temel tip kendi altlarında farklı tipler barındırır.

Nesneye yönelimli programlama dilinin temeli sınıf(class) mantığına dayanır ve java da veri tipi olarak sınıflarımız vardır. Bu veri tiplerinden bahsedecek olursak bizim ilkel(primitive) veri tiplerimizin birer tane gömüldüğü sınıf vardır. Bunlar Byte, Short, Integer, Double, Character, Float, Double, Boolean sınıflarıdır peki diyebiliriz ki ilkel veri tipleri varsa neden sınıf olan veri tiplerine ihtiyacımız var ya da sınıf veri tiplerimiz varsa neden ilkel veri tiplerine ihtiyacımız var bunların farkı nelerdir. İlkel veri tipleri ile sınıf veri tiplerinin farkını int ilkel tipi ile Integer veri tipi arasında ki farkı göstererek anlatalım;

int bir primitif veri tiptir Integer ise bir nesnedir

Integer bir değişken olduğu için null değeri verilebilir ama int'e verilmez. (defaultu 0)

Integer tipinde bir değişken java.lang.Integer sınıfında ki metotları kullanabilir, int ise kullanamaz.

Integer tipinde ki değişkenleri bir vektör ya da koleksiyon içerisinde tutabiliriz ama int ile bunu yapamayız.

Primitif veri tipleri bir nesne olmadığı için serialize (serileştirme) işlemine tutamayız.

Integer gibi sınıflar immutable (değişmez) özelliğe sahiptirler, sıradan nesneler gibi davranmazlar.

Aynı benzer durumlar byte-Byte,short-Short,int-Integer,float-Float,double-Double,char-Character,boolean-Boolean için de geçerlidir.

NOT : Java derleyicisi gerektiğinde tanımlanan her ilkel veriyi ait olduğu sınıfa otomatik olarak gömer buna kutulama (boxing) denilir ya da derleyici gerektiğinde bir sınıf nesnesini ilkel veri tipine döndürür buna da kutu açma (unboxing) denilir.Her iki işlemi de Java derleyicisi kendiliğinden otomatikman yapar. Boxing, autoboxing, unboxing , autounboxing gibi bazı özelliklere de kısaca bakalım.

```
Integer boxing = Integer.valueOf(10);

Integer autoboxing =10;

int unboxing = boxing.intValue();

int autounboxing=10;
```

Peki wrapper class nedir ? Büyük harfle kullandığımız ifadeler yani Byte, Short, Integer, Long, Double,, Float, Character, Boolean sınıfları primitive veri tiplerinin Wrapper (Kapsayıcı) sınıfları olarak ifade edilir ve referans tipindedidir.

Not : Primitive Veri Tipleri değerleri stack (yığın) üzerinde tutulurken Referance Veri Tipleri değerleri heap(yığın) üzerinde tutulur.

28.C

Java'da **int, ilkel bir veri türüdür, Integer ise bir Wrappersınıfıdır**.

- int, ilkel bir veri türü olmak daha az esnekliğe sahiptir. İçinde yalnızca bir tamsayının ikili değerini saklayabiliriz.
- Integer, int veri türü için bir wrapper class içinde olduğundan, bize bir int verisini depolama, dönüştürme ve işleme konusunda daha fazla esneklik sağlar.
- Integerbir sınıftır ve bu nedenle sınıfta tanımlanan metotları çağırabilir. Integer türünün değişkenleri, diğer tüm referans (nesne) türlerinde olduğu gibi, Integer nesnelerinin referanslarını depolar.

```
publicclass Q28 {
	publicstaticvoid main(String[] args) {
		// TODO Auto-generatedmethodstub
		Integerinteger = newInteger(4);
		System.out.print(integer.byteValue());
		System.out.print("-");
		int i = newInteger(4);
		System.out.print(i.byteValue());
		// primitive olan i değişkeni içinde olmayan bir meota erişilmeye çalışılıyor.
	}
}
```

29.B

java da method çağırımı yapılırken MetotIsmi() seklinde yapılır. “newmetotIsmi” veya “newMetotIsmi()” yapıları bir obje oluştururken kullanılan yapılardır.

30.A

```
1	String cat = "animal", dog = "animal";
2	String cat = "animal"; dog ="animal";
3	String cat, dog ="animal";
4	String cat, String dog = "animal";
```

2’de dog değişkenin türü belirtilmemiş.
3’de cat değişkeni initialize edilmemiş.
4’de cat değerine atama yapılmammış.

31.C

byte, short, int, long, double, float, char, boolean

Byte, Short, Integer, Long, Double,, Float, Character, Boolean

32.B

Soru 25 te açıklanan bilgilere göre instance variable özellikleri incelendiğinde B seceneği dogrudur.

33.A

- İlkel tür, Java tarafından sağlanan önceden tanımlanmış bir veri türüdür.
- İlkel tür bir nesne değildir, dolayısıyla bir sınıfa ait değildir.
- İlkel bir veri türü null değerlere izin vermez.

- Gerekli bellek Wrapper sınıflarına göre daha düşük.
- Koleksiyonlarda ilkel tip kullanılmaz.

34.B

Kodlama içerisinde Full GC çalıştırmak için System.gc(); çağırmanız yeterli olacaktır.

35.C

İşlemler sonunda fruit1,fruit2, fruit3 hepsi “apple” oluyot. O yüzden fruit2 ve fruit3 ün tutulmasına gerek yoktur.

36.B

```
double d = new Double(1_000_000.00);
```

37.B

```
public class Q37 {
	public String first = "instance";

	public Q37() {
		first = "constructor";
	}

	{  // instance initialization block”  obje her instantiate edildiğinde çalısır.

		first = "block";
	}

	public void print() {
		System.out.println(first);
	}

	public static void main(String... args) {
		new Q37().print();	}
}
```

38.C

int, ilkel yapida oldugu icin null degeri almaz. Integer ve String ifadeleri wrapper calslardan yaratilmis objeler olduklari icin null degerleri alirlar.

39.B

Statik blank sadece statik methotta olurken instance blank hem statikte hem de instance de olabilmektedir.

```
public class Q15 {

	{		//illustrate instance initializers
		System.out.println();
	}

	public Q15() {		//constructor
		System.out.println();
	}

	static {	//static initializers
		System.out.println();
	}
	{		//illustrate instance initializers
		System.out.println();
```

```
        }
}
```

40.B

Sayısal ifadelerde _ kullanılabilir ama ayısal ifade başında ve virgülden sonraki ilk kısımda kullanılamaz.

41.A

Bayt veri türü, 8 bit verileri saklamak için kullanılır.

Short veri türü,16 bit verileri saklamak için kullanılır.

int veri türü 32- bit verileri saklamak için kullanılır.

long veri tipi 64- bit verileri saklamak için kullanılır.

42.A

```
public class Q42 {
        public String name;

public static void main(String[] meow) {
        Q42 cat = new Q42();
         cat.name = "Sadie"; //class içindeki name değişkenine ulaşılmış
}
}
```

43.B ???GC

```
public class Q43Toy {
        public void play() {
                System.out.print("play-");
        }

        public void finalizer() {
                System.out.print("clean-");
        }

        public static void main(String[] fun) {
                Q43Toy car = new Q43Toy();
                car.play();
                System.gc();
                Q43Toy doll = new Q43Toy();
                doll.play();
        }
}
```

44.D

45.A

Integer olması için new kelimesinin kullanılması gerekir.

```
                int first = Integer.parseInt("5");
                int second = Integer.valueOf("5");
```

46.D

Kod sonunda elena, diana, zoe isimli objelerin hepsi null olduğu için silinebilir.

47.C

Constructor public olmalıdır ve sınıf ismiyle aynı olmalıdır.

48.D ???GC

49.B

50.C

NOT: Stack ve Heap kavramları için link : http://blog.bilgiyazan.com.tr/stack-ve-heap-kavrami/