- In the following code, what is the printout for list2?

```
class  Test {
  public static void main(String[] args) {
    int[] list1 = {3, 2, 1};
    int[] list2 = {1, 2, 3};
    list2 = list1;
    list1[0]   = list1[0] * list2[0];
    list1[1]  = list1[1] * list2[1];
    list1[2]  = list1[2] * list2[2];
    for(int i = list2.length – 1; i >= 0; i--)
         System.out.print(list2[i] + " ");
```

- If you declare an array double[] list = {3.4, 2.0, 3.5, 5.5}, list[1] is _____.

```
}
```

- Write the following method that returns true if the list is already sorted in increasing order.

```
public static boolean isSorted(int[] list)
```

<Output>
Enter list: 8 10 1 5 16 61 9 11 1 The list is not sorted
<End Output>

<Output>
Enter list: 10 1 1 3 4 4 5 7 9 11 21 The list is already sorted

- Write a method that computes the number of letters in a String:

The method signature is as follows:

```
Public int count(String s)
{


}
```

- Write a method called *count* that counts number of occurrences of all odd elements in an array of integer values. For example, if:

List

5	10	15	5	30	43	5	60	70	80

Then coun(list) will return 5

```
public static int count(int[] list)
{


}
```

1. _____ describes the state of an object.
    a. data fields
    b. methods
    c. constructors
    d. none of the above

2. An attribute that is shared by all objects of the class is coded using _____.

    a. an instance variable
    b. a static variable
    c. an instance method
    d. a static method

3. If a class named Student has no constructors defined explicitly, the following constructor is implicitly provided.

    a. public Student()
    b. protected Student()
    c. private Student()
    d. public Student(String name)

4. If a class named Student has a constructor Student(String name) defined explicitly, the following constructor is implicitly provided.

a. public Student()
b. protected Student()
c. private Student()
d. Student()
e. None

5. Suppose the xMethod() is invoked from a main method in a class as follows, xMethod() is _____ in the class.

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

  xMethod();
}
```

a. a static method
b. an instance method

6. What would be the result of attempting to compile and run the following code?

```
public class Test {
  static int x;

  public static void main(String[] args){
          System.out.println("Value is " + x);
  } }
```

a. The output "Value is 0" is printed.
b. An "illegal array declaration syntax" compiler error occurs.
c. A "possible reference before assignment" compiler error occurs.
d. A runtime error occurs, because x is not initialized.

7. Analyze the following code:

```
public class Test {
   private int t;

  public static void main(String[] args) {
    Test test = new Test();
    System.out.println(test.t);
 } }
```

a. The variable t is not initialized and therefore causes errors.
b. The variable t is private and therefore cannot be accessed in the main method.
c. Since t is an instance variable, it cannot appear in the static main method.
d. The program compiles and runs fine.

8. How can you get the word "abc" in the main method from the following call?

```
java Test "+"  3  "abc"  2
```

a. args[0]
b. args[1]
c. args[2]

d. args[3]

9. What is the output of running class C?

```
class A {
  public A() {
    System.out.println("The default constructor of A is invoked");
  } }

class B extends A {
  public B() {
   System.out.println("The default constructor of B is invoked");
  } }

public class C {
  public static void main(String[] args)  {
    B b = new B();
 } }
```

a. none
b. "The default constructor of B is invoked"
c. "The default constructor of A is invoked" followed by "The default constructor of B is invoked"
d. "The default constructor of A is invoked"

10) Which members of base class `Players` are inherited by `SoccerPlayers`?

```
public class Players {
      public void setName(String newName) { … }
      public void setAge(int newAge) { … }
      public void printDetails() { … }
      private String playerName;
      private int playerAge;
};


public class SoccerPlayers extends Players {
      public void setDetails(String newName) { … }
      public String getLeague() { … }
      private String teamName;
};
```

a. only playerName and playerAge

b. only setName, setAge, printDetails
c. setDetails, getLeague, playerName, playerAge, and teamName
*d. setName, setAge, printDetails, playerName, and playerAge

11) Deriving properties of a base class is called _____.
a. extension
*b. inheritance
c. overloading
d. implementation

12) Which statement in `main()` generates an error?

```
public class Players {
      public void setName(String newName) { … }
      public void setAge(int newAge) { … }
};
public class SoccerPlayers extends Players {
      public void setDetails(String newName) { … }
      public String getLeague() { … }
};
public static void main(String args[]) {
   String leagueName;
   Players newPlayers = new Players();
   SoccerPlayers playerObject = new SoccerPlayers();
   playerObject.setName("Jessica Smith");
   playerObject.setAge(21);
   leagueName = newPlayers.getLeague();
}
```

a. `Players newPlayers = new Players();` An object of a base class cannot be created
b. `playerObject.setName("Jessica Smith");` setName() is not declared in SoccerPlayers
*c. `leagueName = newPlayers.getLeague();` getLeague() is not a member of Players
d. `SoccerPlayers playerObject = new SoccerPlayers();` An object of a derived class cannot be created.

13) Which is true?

```
public class Vehicle {
   protected String name;
   private int ID;
}
public class Car extends Vehicle {
   protected int miles;
}
```

a. Car members have access to Vehicle ID
b. Vehicle members have access to Car miles

c. Car members do not have access to any Vehicle members
*d. Car members have access to Vehicle name

14) If class Animal can only be accessed by classes defined in the same package as Animal, which is the appropriate class definition?
a. public class Animal
*b. class Animal
c. private class Animal
d. protected class Animal

7) Which is true?

```
public class Person {
   String name;
}
```

a. Name can be accessed only by members of Person
b. Name can be accessed by members of any class
c. Name can be accessed by only derived classes of Person
*d. Name can be accessed by members of Person and other classes in the same package

15) Which lines cause a compile error?

```
public class Vehicle {
   protected String name;
   private int ID;
   public setID(int pID) {...}
}
public class Car extends Vehicle {
   private int miles;
   public Car() {
1     miles = 145;
2     setID(99);
3     ID = 47;
4     name = "Honda";
   }
}
```

a. lines 1 and 3
b. there are no errors
c. lines 3 and 4
*d. line 3 only

16) Declaring a member as ____ in the base class provides access to that member in the derived classes but not to anyone else.
a. public
*b. protected
c. private

d. constant

17) Given class`SimpleCar`, which line has a syntax error?

```
public class SimpleCar {
   private int odometer;
   public void drive(int miles) {
      odometer = odometer + miles;
   }
}
1 Object objCar;
2 objCar = new SimpleCar();
3 System.out.println(objCar.toString());
4 objCar.drive();
```

a. line 1 - class Object is not defined
b. line 2 - a SimpleCar reference cannot be assigned to an Object reference
c. line 3 - toString( ) is not defined in class SimpleCar
*d. line 4 - drive( ) is not defined in class Object

18) Which XXX allows a collection of Vehicle Objects to dynamically invoke drive() in `Car`and`Boat`?

```
public class Vehicle {
   XXX
}
public class Plane extends Vehicle {
   @Override
   public void drive() {...}
}
public class Boat extends Vehicle {
   @Override
   public void drive(){...}
}
```

*a. public void drive() {...}
b. private void drive() {...}
c. public void drive(int miles) {...}
d. public void super.drive() {...}

19) What is output?

```
import java.util.ArrayList;
public class SimpleCar {
   @Override
   public String toString(){
      return "I drive fast";
   }
   public static void main(String[] args){
```

```
        ArrayList <Object> myStuff;
        myStuff = new ArrayList<Object>();
        myStuff.add(new String("Greetings"));
        myStuff.add(new Object());
        myStuff.add(new SimpleCar());
        for(Object item : myStuff){
           System.out.println(item.toString());
        }
    }
}
```

a. String
Object
SimpleCar
b. java.lang.Object@19cc
java.lang.Object@23fb
java.lang.Object@ab79
c. java.lang.String@169b
java.lang.Object@23fb
java.lang.SimpleCar@a42b
*d. Greetings
java.lang.Object@169b
I drive fast

20) A ___ encapsulates data and behavior to create objects.
a. method
*b. class
c. UML diagram
d. header file

21) Consider a Dog class and a Mammal class. Which is true?
a. The Dog class is abstract
b. The Mammal class is concrete
c. Both classes are concrete
*d. The Mammal class is abstract

22) A(n) ___ guides the design of subclasses but cannot be instantiated as an object.
a. child class
b. base class
*c. abstract class
d. derived class

23) Programmers often use a powerful programming paradigm that consists of three key features — classes, inheritance, and abstract classes. What is the paradigm called?
a. Modular programming
b. Structured programming
c. Universal Modeling Language
*d. Object-oriented programming

24) Which is true?
a. Concrete classes do not have any member methods.
b. Concrete classes cannot be instantiated.
c. Concrete classes cannot be inherited.
*d. Concrete classes do not have any abstract methods.

25) What is output?

```
public abstract class People {
   protected String name;
   protected int age;
   public abstract void PrintInfo();
   public void PrintInformation() {
      System.out.println("In Base Class People");
   }
}
public class Teacher extends People {
   private int experience;
   public void PrintInfo() {
      System.out.println("In Child Class Teacher");
   }
}
public class Principal extends Teacher {
   public void PrintInformation() {
      System.out.println("In Child Class Principal");
  }
   public static void main(String args[]) {
      Principal tim;
      tim = new Principal();
      tim.PrintInfo();
   }
}
```

a. In Child Class Principal
*b. In Child Class Teacher
c. In Base Class People
d. Error: Compiler error

26) What is output?

```
public abstract class Vehicle {
   public abstract void move(int miles);
   public void printInfo(){
      System.out.print("Vehicle ");
   }
}
public class Car extends Vehicle {
   private int distance;
   public void move(int miles) {
      distance = distance + miles;
   }
   public void printInfo() {
      System.out.print("Car ");
   }
   public static void main(String args[]) {
      Vehicle myVehicle;
      Car myCar;
      myVehicle = new Vehicle();
      myCar = new Car();
      myVehicle.printInfo();
      myCar.printInfo();
   }
}
```

a. Vehicle
b. Vehicle Car
c. Error: Car is not abstract and does not override abstract method move()
*d. Error: Vehicle is abstract and cannot be instantiated

Review

Money.java, Length.java
Algorithms

isPrime(int n)
selectionSort(int[] list)
binarySearch(int[] list, int key)
linearSerach(int[] list, int key)
isPalindrome(String s)
arrays (find max,find avg, reverse)

generate a list of lowerCase/upperCase letters

- Tanks containing some fluid are described by the following class

```
public  class Tank
{
        private int capacity;    // capacity of the Tank
        private int level;       // level of the fluid in the Tank (must be <= capacity

        // Create an empty Tank with capacity of c
        public  Tank(int c)
        {
                capacity = c;
                level = 0;
        }

        // Fill up this tank to capacity
        public  void  fill()
        {
                level = capacity;
        }

        // Empty out this Tank
        public  void empty()
        {
                Level = 0;
        }

        // Get the capacity of this Tank
        public  int getCapacity()
        {
                return capacity;
        }

        // get the level of this Tank
        public int getLevel()
        {
                return level;
        }

// Pour fluid from other Tank into this tank
        public  void pourFrom(Tank other)
        {

        }
}
```

*Implement the method pourFrom(Tank other), which pours liquid from Tank other into Tank this. For example, the call t1.pourFrom(t2) should pour all of the liquid from t2 into t1 unless this would cause Tank t1 to overflow. In that case, t1.pourFrom(t2) should fill up Tank t1 and leave the remainder of the liquid in Tank t2.

```
 public void pourFrom(Tank other)
{


}
```

Given two tanks with capacity of 7 and 4, write a complete java program TestTank.java to produce 5 units of fluid in larger tank. The algorithm is:

Create the smaller tank
Create the larger tank
Fill the smaller tank
Pour it into the larger tank
Fill the smaller tank again
Pour into larger one till filled
Empty the larger tank
Pour the smaller tank into larger tank
Fill the smaller tank
Pour smaller tank into larger tank