Q1 Write a Java program that reads 10 integers from the keyboard and outputs all the pairs whose sum is 30.

Solution:

```java
import java.io.*;
import java.util.Scanner;
public class Question1 {
    public static void main(String args[]) throws IOException {
        int i, j, a[];
        a = new int[10];
        Scanner s = new Scanner(System.in);
        for (i=0; i<10; i++) {
            System.out.print("Please enter an integer: ");
            while (!s.hasNextInt()) {
                s.nextLine();
                System.out.print("That's not an integer; please enter an integer: ");
            }
            a[i] = s.nextInt();
        }
        for (i=0; i<9; i++)
            for (j=i+1; j<10; j++)
                if (a[i]+a[j]==30)
                    System.out.println(a[i]+"+"+a[j]+"=30");
    }
}
```

Q2 Write a Java program that takes two arrays a and b of length n storing int values, and returns the dot product of a and b. That is, it returns an array c of length n such that c[i]=a[i]*b[i].

Solution:

```java
class ArraySizeException extends Exception {
    public ArraySizeException() { super(); }
    public ArraySizeException(String s) { super(s); }
}
public int[] compute(int[] a, int[] b) throws ArraySizeException {
    if (a.length != b.length) {
        throw new ArraySizeException("arrays must have same length");
    }
    int[] c = new int[a.length];
    for (int i = 0; i < a.length; i++) {
        c[i] = a[i] * b[i];
    }
    return c;
}
```

Q3 Explain why the Java dynamic dispatch algorithm, which looks for the method to invoke for a call o.a(), will never get into an infinite loop.

Solution: Inheritance in Java allows for specialized classes to be built from generic classes. Because of this progression from generic to specialized in the class hierarchy, there can never be a circular pattern of inheritance. In other words, there cannot be a superclass A and derived classes B and C such that B
extends A, then C extends B, and finally A extends C. Such a cycle is impossible because A is the
generic superclass from which C is eventually extended, thus it is impossible from A to extend C, for
this would mean A is extending itself. Therefore, there can never occur a circular relationship which
would cause an infinite loop in the dynamic dispatch.

Q4 Consider the following code segment, taken from some package:

```java
public class Maryland extends State {
    Maryland() { /* null constructor */ }
    public void printMe() { System.out.println("Read it."); }
    public static void main(String[] args) {
        Region mid = new State();
        State md = new Maryland();
        Object obj = new Place();
        Place usa = new Region();
        md.printMe();
        mid.printMe();
        ((Place) obj).printMe();
        obj = md;
        ((Maryland) obj).printMe();
        obj = usa;
        ((Place) obj).printMe();
        usa = md;
        ((Place) usa).printMe();
    }
}
class State extends Region {
    State() { /* null constructor */ }
    public void printMe() { System.out.println("Ship it."); }
}
class Region extends Place {
    Region() { /* null constructor */ }
    public void printMe() { System.out.println("Box it."); }
}
class Place extends Object {
    Place() { /* null constructor */ }
    public void printMe() { System.out.println("Buy it."); }
}
```

What is the output from calling the main() method of Maryland class?

**Solution:**

Read it.
Ship it.
Buy it.
Read it.
Box it.
Read it.

Q5 Write a program that consists of three classes, A, B, and C, such that B extends A and C extends B.
each class should define an instance variable named "x" (that is, each has its own variable named x).
Describe a way for a method in C to access and set A's version of x to a given value, without changing
B or C's version.

**Solution:**

```java
public class A {
    int x = 1;
```
public void setIt(int y) { x = y; }
public int getIt() { return x; }
}
public class B extends A {
  int x = 2;
  public void setIt (int y) { x = y; }
  public int getIt() { return x; }
  public void superSetIt (int y) { super.x = y; }
  public int superGetIt() { return super.x; }
}
public class C extends B {
  int x = 3;
  public void setIt (int y) { x = y; }
  public int getIt() { return x; }
  public void superSetIt (int y) { super.x = y; }
  public int superGetIt() { return super.x; }
  public void superDuperSetIt(int y) { super.superSetIt(y); }
  public int superDuperGetIt() { return super.superGetIt(); }
  public static void main(String[] args) {
    C c = new C();
    System.out.println("C's is" + c.getIt());
    System.out.println("B's is" + c.superGetIt());
    System.out.println("A's is" + c.superDuperGetIt());
    c.superDuperSetIt(4);