Questions 14

```
public class Question_14
{
   public static void main(String[] args)
   {
      boolean areReversed = reverseSame(1234, 4321);
      System.out.println(areReversed);

      System.out.println( reverseSame(9876, 4321) );
   }

   private static boolean reverseSame(int num1, int num2)
   {
  // To determine whether num1 and num2 contain the same digits in reverse order,
  // this method reverses the digits in num1 into num1Reversed and then
  // compares num1Reversed against num2.
  // An example of digit-reversing process: num1 = 1234:
  // num1: 1234 123 12 1 0
  // num1Reversed: 0 4 43 432 4321
      int num1Reversed = 0;
      while( num1 != 0 )
      {
         // Extract the rightmost digit of num1
         int rightmostDigit = num1 % 10 ;
         // and make it the rightmost digit in num1Reversed.
         num1Reversed = 10 * num1Reversed + rightmostDigit;
         num1 = num1 / 10; // Drop the rightmost digit of num1
      }
      return num1Reversed == num2;
   }
}
```

Question 15

```
import java.util.Arrays;

public class Product
{
   private long upc;
   private String name;
   private double price;
   private int quantityOnHand;
   private boolean[] onSpecial = new boolean[52];

   public Product(long upc, String name, boolean[] onSale)
   {
      this.upc = upc;
      this.name = name;
      this.price = 0.0;
      this.quantityOnHand = 0;
// this.onSpecial = onSale; // Shallow copy no good!
// Deep copy very good!
      for(int week = 0; week < onSpecial.length && week < onSale.length; ++week )
      {
         this.onSpecial[week] = onSale[week];
      }
   }

   public boolean outOfStock()
   {
      return quantityOnHand == 0;
   }

   // Overload
   public boolean equals(Product other)
   {
        if (!name.equals(other.name))
         return false;

      if (price != other.price)
         return false;

      if (quantityOnHand != other.quantityOnHand)
         return false;

      if (upc != other.upc)
         return false;
```

```
   // Normally, here is how you compare two arrays of primitive types
   // if (!Arrays.equals(onSpecial, other.onSpecial))
   // But if you must reinvent the wheel:

   // check the length first
   if(onSpecial.length != other.onSpecial.length)
      return false;

   // Now, that the array lengths are the same, compare their elements
   for(int week = 0; week < onSpecial.length; ++week )
   {
      if( this.onSpecial[week] != other.onSpecial[week] )
         return false;
   }
   return true;
}

public void addStock(int quantity)
{
   this.quantityOnHand += quantity;
}

public String getName()
{
   return name;
}


public static void main(String[] args)
{
   // Q15,c
   boolean[] onSale = new boolean[52]; // all set to false by default
   Product[] store = new Product[50];
   for(int product = 0; product < store.length ; ++product )
   {
      String productName = "Product" + (product+1);
      store[product] = new Product(0, productName, onSale);
   }

   store[0].setPrice(50);
   for(int product = 1; product < store.length ; ++product )
   {
      store[product].setPrice(50 + store[product-1].getPrice());
   }
```

```
      // Q15,d
      for(int product = 0; product < store.length ; ++product )
      {
         System.out.println(store[product].getName());
      }

      // Q15,e
      store[store.length-1].addStock(50);

   }

// we need this method in Q15,c
   public double getPrice()
   {
      return price;
   }

   // we need this method in Q15,c
   public void setPrice(double price)
   {
      this.price = price;
   }
}
```

Question 16

```
class Pair
{
   int first, second;

   public Pair() { first = second = 0; }
   public Pair(int f, int s) { first = f; second = s; }
   int getFirst() { return first; }
   int getSecond() { return second; }
   void setFirst(int f) { first = f; }
   void setSecond(int s){ second = s; }
   void printPair()
   {
      System.out.println("Pair: " + first + ", " + second);
   }
}
```

```
public class PairTest
{
   public static void main(String[] args)
   {
      int i = 5;
      Pair c1 = new Pair();
      Pair c2 = new Pair(2, 3);
      c1.printPair();
      c1.setSecond(i);
      c1.printPair();
      c2.printPair();
      c2.setFirst(i * 3);
      c2.printPair();
      System.out.println(c1.getFirst());
      Pair c3 = c2;
      c3.setFirst(i * 4);
      c2.printPair();
      c3.printPair();
      if (c2 == c3)
         System.out.print("here 1 ");
      c3 = new Pair(4, 3);
      if (c2 == c3)
         System.out.print("here 2 ");
   }
}
```

```
Output
_____________________
Pair: 0, 0
Pair: 0, 5
Pair: 2, 3
Pair: 15, 3
0
Pair: 20, 3
Pair: 20, 3
here 1
_____________________
```

Question 17

```
import java.util.Scanner;
public class Hour_Glass_Figure
{
   public static void main(String[] args)
   {
      draw_hour_glass_figure();
   }
   public static void draw_hour_glass_figure()
   {
      Scanner keyboard = new Scanner(System.in);
      System.out.print("Enter a number? ");
      int number = keyboard.nextInt();
// Note: this implementation is independent of whether number is even or odd

      // Print the top half
      int leadingSpaces = 0; // initial number of leading spaces for the top half
      int stars; // number of stars to be printed
      for( stars = number; stars > 0 ; stars -= 2 )
      {
         // print leading spaces
         for(int ls = 1; ls <= leadingSpaces ; ++ls ) { System.out.print(" "); }
         // print stars
         for(int s = 1; s <= stars ; ++s ) { System.out.print("*"); }
         System.out.println(); // skip line
         ++leadingSpaces; // Increment number of leading spaces for the next line
      }

      // Print the bottom half
      --leadingSpaces; // Undo the last increment of the number of leading spaces
      --leadingSpaces; // initialize the number of leading spaces for the bottom half
      for(stars += 4; stars <= number ; stars += 2 )
      {
         // print leading spaces
         for(int ls = 1; ls <= leadingSpaces ; ++ls ) { System.out.print(" "); }
         // print stars
         for(int s = 1; s <= stars ; ++s ) { System.out.print("*"); }
         System.out.println(); // skip line
         --leadingSpaces; // Decrement number of leading spaces for the next line
      }
   }
}
```