Puzzle 1: Strange Loops

java

```
public class StrangeLoops {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            System.out.println(i);
            i = i++;
        }
    }
}
```

Question: What will the output of the above program be?

Answer: The output will be 0 1 2 3 4 5 6 7 8 9.

Explanation: The expression i = i++ seems to imply that i is incremented and then reassigned its old value. However, due to how post-increment works in Java, i++ actually increments i, but the assignment has no effect since it reassigns i to its value before the increment. Thus, the increment still takes effect, and the loop behaves as if i++ were simply i++.
Puzzle 2: String Comparison

java

```
public class StringComparison {
    public static void main(String[] args) {
        String s1 = "Hello";
        String s2 = "Hello";
        String s3 = new String("Hello");

        System.out.println(s1 == s2);
        System.out.println(s1 == s3);
        System.out.println(s1.equals(s3));
    }
}
```

Question: What will the output of the above program be?

Answer: The output will be:

arduino

```
true
false
true
```

Explanation:

s1 == s2 is true because both s1 and s2 refer to the same string literal in the string pool.
s1 == s3 is false because s3 is a new String object, and the == operator compares object references, not values.
s1.equals(s3) is true because the equals method compares the actual contents of the strings.

Puzzle 3: Autoboxing and Unboxing

java

```
public class Autoboxing {
    public static void main(String[] args) {
        Integer a = 1000;
        Integer b = 1000;

        System.out.println(a == b);

        Integer c = 100;
        Integer d = 100;

        System.out.println(c == d);
    }
}
```

Question: What will the output of the above program be?

Answer: The output will be:

arduino

```
false
true
```

Explanation:

a and b are different objects because autoboxing does not guarantee caching for values outside the range -128 to 127. Hence, a == b is false.
c and d refer to the same cached Integer object for values between -128 and 127, so c == d is true.

Puzzle 4: Exception Handling

java

```
public class ExceptionHandling {
   public static void main(String[] args) {
      try {
         int[] arr = new int[10];
         System.out.println(arr[10]);
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Array index out of bounds!");
      } catch (Exception e) {
         System.out.println("Some other exception");
      } finally {
         System.out.println("Finally block executed");
      }
   }
}
```

Question: What will the output of the above program be?

Answer: The output will be:

sql

```
Array index out of bounds!
Finally block executed
```

Explanation:

An ArrayIndexOutOfBoundsException is thrown because arr[10] is out of bounds.
The first catch block catches the exception, and "Array index out of bounds!" is printed.
The finally block is always executed, so "Finally block executed" is printed.

Puzzle 5: Integer Caching

java

```
public class IntegerCaching {
   public static void main(String[] args) {
      Integer a = 127;
      Integer b = 127;
      Integer c = 128;
      Integer d = 128;

      System.out.println(a == b);
      System.out.println(c == d);
   }
}
```

Question: What will the output of the above program be?

Answer: The output will be:

arduino

true
false

Explanation:

a and b refer to the same cached Integer object for the value 127.
c and d do not refer to the same object because the value 128 is outside the range of the cache.