**Name**

**Advanced Programming in Java**

**Lab Exercise 10/29/2020**

**Wrapper Classes**

Reference: Lesson 21 in Blue Pelican Java

1. The classes that convert primitives to objects are called \_\_\_\_\_\_\_\_\_\_\_\_\_classes.

2. Name the four primitive data types with which wrapper classes primarily deal.

3. Write code that will convert *double dx* into a wrapper class object. Call the object *dd*.

4. Write code that will produce a *Boolean* type wrapper object called *bj* (“wrap” a *true*

inside it).

5. Write code that will convert the integer *ip* into an *Integer* wrapper class object. Call the

object *ozzie*.

6. Assume you have the object *Character cw*. Write code to convert this to a primitive

character.

7. Assume you have *Double* objects *d1* and *d2*. Show how you would multiply the values

stored in these objects and then store the answer in primitive *double dd*.

8. Assume you have *Integer* objects *i1* and *i2*. Show how you would add the values stored

in these objects and then store the answer in a third *Integer* object called *i3*.

9. Write code that will extract the *boolean* wrapped in the *Boolean wnOh* and test it with an *if* statement.

10. Convert the object *jj* (of type *Double*) into a primitive *float* called *ff*.

11. Convert the object *pk* (of type *Double*) into a primitive *int* called *gurg*. What is the

danger of doing this?

12. What is the primary purpose of wrapper classes?

**Programming Problem**

1. Write a program that allows the user to enter two floating point values using the Scanner class. The program should then store them in a Double wrapper class. The program should then extract the values stored in the two Double class objects and add them together storing the result in a Double wrapper class object. The program should then take the square root of the value stored in the Double wrapper class object holding the sum of the two input values.
2. While numbers do not feel emotion, people do respond emotionally when numbers behave consistently in surprising and amusing ways. Happy Numbers can be reduced to 1 with a simple formula.

To determine if a number is happy, or not, follow these simple rules with any number:

* If the number is a single digit, square it.
* If the result or the original number has multiple digits, take each digit by itself and square the digit and add the squares of the digits.
* Repeat until you get to the number 1 or you find the results are repeating.

Happy numbers, and other numeric patterns, can be identified programmatically with code. The first step to creating pseudo code to evaluate happy numbers is to remind yourself how to determine if a number is happy or not. You take any number, break the number into single digits, then square each digit and add up the sum of all you multiplications. You repeat this process until either you get the number 1 (the number you started with is happy) or your results match a pattern that indicates an unhappy number (the sequence of results starts repeating).

Your task is to write a program that will find all of the happy numbers from 1 to 1000. I recommend writing a function isHappy that is passed an integer and returns True if the integer is happy and False if the number is sad.

1. In number theory, a perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, 6 has divisors 1, 2 and 3 (excluding itself), and 1 + 2 + 3 = 6, so 6 is a perfect number. Write a program that will report all of the perfect numbers less than or equal to 10000. Do you see any pattern in the perfect numbers?

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What are the perfect numbers <= 10000?

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. In mathematics, a **Mersenne prime** is a prime number that is one less than a power of two. That is, it is a prime number of the form *Mn* = 2*n* − 1 for some integer *n*. They are named after Marin Mersenne, a French Minim friar, who studied them in the early 17th century. Specifically, A Mersenne prime is a Mersenne number, i.e., a number of the form

|  |
| --- |
| M_n=2^n-1, |

that is prime. In order for M_n to be prime, n must itself be prime. It should be noted that there are only 51 known Mersenne primes. The last 17 were found by GIMPS (Great Internet Mersenne Primes Search) using distributed computing techniques. The 8th one was found by Leonard Euler in 1772. The 51st one was found in 2018 with a value of 282589933 – 1 (24862048 digits).

Write a program that will report the first 8 Mersenne primes

**Submit this sheet as well as the documented source code.**