Morning Katas

Day 12

Print Your Name

Create a new Console Application named `Day12.Kata1`

In the `Main` method, `WriteLine` your name

Print Your Name

```
public static void Main(string[] args)
{
  Console.WriteLine("Curtis");
}
```

Declare and Use a Variable

Create a new Console Application named
`Day12.Kata2`

In the `Main` method:
Declare an `int` variable named **`day`** and set it to 12

`WriteLine` the message *"Today is day 12"* using the **+** operator to concatenate the **`string`** *"Today is day"* and the **`day`** variable

Declare and Use a Variable

```
public static void Main(string[] args)
{
   int day = 12;
   Console.WriteLine("Today is day " + day);
}
```

Use an `if-else`

Create a new Console Application named
`Day12.Kata3`

In the `Main` method:
Declare a `new` instance of `Random`
Use the `Next(10)` method to get the next `int`
If that value is less than 5, `WriteLine` *"Less than 5"*
Otherwise, `WriteLine` *"Greater than or equal to 5"*

Use an `if-else`

```
public static void Main(string[] args)
{
   Random r = new Random();
   if (r.Next(10) < 5)
   {
      Console.WriteLine("Less than 5");
   }
   else
   {
      Console.WriteLine("Greater than or equal to 5");
   }
}
```

Use an array

Create a new Console Application named
`Day12.Kata4`

In the `Main` method:
Declare and initialize an array of `string`s that contain the names of names of the week
`WriteLine` the entry “Tuesday”

Use an array

```
public static void Main(string[] args)
{
   string[] daysOfTheWeek = new string[]
   {
      "Sunday", "Monday", "Tuesday", "Wednesday",
      "Thursday", "Friday", "Saturday"
   };
   Console.WriteLine(daysOfTheWeek[2]);
}
```

Use the `for` loop

Create a new Console Application named
`Day12.Kata5`

In the `Main` method:
Declare and initialize an array of three random `int`s less than 100
Declare a variable named `total`
Use a `for` loop to loop over the array and sum the total
`WriteLine` the total

Use the `for` loop

```
public static void Main(string[] args)
{
  Random r = new Random();
  int[] nums = new int[]
  {
    r.Next(100), r.Next(100), r.Next(100)
  };
  int total = 0;
  for (int i = 0; i < 3; i = i + 1)
  {
    total = total + nums[i];
  }
  Console.WriteLine(total);
}
```

Handle User Input

Create a new Console Application named
`Day12.Kata6`

In the `Main` method:
Write a message that asks the user for their name
Get their name from the input
Write the message *“Hello, «name»!”*

Handle User Input

```
public static void Main(string[] args)
{
  Console.WriteLine("What is your name?");
  string name = Console.ReadLine();
  Console.WriteLine("Hello, " + name);
}
```

Declare a Class & Method

Create a new Console Application named
`Day12.Kata7`

Create a class named `BankAccount` with a constructor that sets the balance to a random number between 0 and 10000. Have a method that returns that balance.

In the `Main` method:
Create an instance of `BankAccount` using `new`.
Print the balance of the bank account.

Declare a Class & Method

```
// In BankAccount.cs
public class BankAccount
{
  private decimal balance;
  public BankAccount()
  {
    Random r = new Random();
    balance = r.Next(10001);
  }
  public decimal GetBalance()
  {
    return balance;
  }
}

// In Program.cs
public static void Main(string[] args)
{
  BankAccount account = new BankAccount();
  decimal balance = account.GetBalance();
  Console.WriteLine("Your balance is: " + balance);
}
```

Child Classes

Create a new Console Application named `Day12.Kata8`

Create an `abstract` class named `Animal` that has a `Speak` method that uses an `abstract` `get`ter property named `Vocalization` of type `string`. and `WriteLine`s the return value of `Vocalization` to the console.

Create a `Fish` class that inherits from the `Animal` class and `override`s the `Vocalization` property by returning “Blurp”.

Create a `HoneyBadger` class that inherits from the `Animal` class and overrides the `Vocalization` property by returning “I OWN THE WORLD!”.

In the `Main` method:
Create a `List<Animal>` and `Add` an instance of `Fish` and `Add` an instance of `HoneyBadger` to it.
Use the `foreach` loop to call the `Speak` method on each instance of `Animal` in the `List<Animal>`

Classes

```
// In Animal.cs
public abstract class Animal
{
  public abstract string Vocalization { get; }
  public void Speak()
  {
    Console.WriteLine(Vocalization);
  }
}
```

Child

```
// In Fish.cs
public class Fish : Animal
{
  public override string Vocalization
  {
    get { return "Blurp"; }
  }
}
```

Child Classes

```
// In HoneyBadger.cs
public class HoneyBadger : Animal
{
  public override string Vocalization
  {
    get { return "I OWN THE WORLD!"; }
}

// In Program.cs, the Main method
public static void Main(string[] args)
{
  List<Animal> farm = new List<Animal>
  {
    new Fish(),
    new HoneyBadger()
  };
  foreach (Animal animal in farm)
  {
    animal.Speak();
  }
}
```

SELECT From a Table

Create a new query

SELECT the description, start date, and end date of all the records from `Sales.SpecialOffer` where the percent discount is greater than 10% and less than 50%

SELECT From a Table

```
SELECT [Description]
     , [StartDate]
     , [EndDate]
  FROM [Sales].[SpecialOffer]
 WHERE [DiscountPct] > 0.10
   AND [DiscountPct] < 0.50
```

SELECT From JOINed Tables

Create a new query

SELECT all of the sales quota and year-to-date sales from the Sales.SalesPerson table and the territory name and the country region for the sales person from the Sales.SalesTerritory table, if they exist

SELECT From JOINed Tables

```
SELECT p.SalesQuota
     , p.SalesYTD
     , t.Name
     , t.CountryRegionCode
  FROM Sales.SalesPerson p
LEFT JOIN Sales.SalesTerritory t
       ON (p.TerritoryID = t.TerritoryID)
```