IdleCash

Links

About

An Open-source Idle Game Currency data type for Unity

How to Install

- Import it from Asset Store
- Import IdleCash.unitypackage from **releases**
- Clone or Download this repository and move to your Unity project's **Assets** folder

Settings

How to Access to Settings Panel

- From Unity Menu Item's `Tools`>`EmreBeratKR`>`IdleCash`>`Settings`

Settings Panel

Default Real Types These letters stands for the **real** value types - `k` : `thousand` - `m` : `million` - `b` : `billion` - `k` : `trillion` - `q` : `quadrillion`

Default Letters These letters stands for the **imaginary** value types

1- Singles - `a`, `b`, `c`, `d`, … , `y`, `z`

2- Doubles - `aa`, `ab`, `ac`, `ad`, … , `ay`, `az` - `ba`, `bb`, `bc`, `bd`, … , `by`, `bz` - … - `ya`, `yb`, `yc`, `yd`, … , `yy`, `yz` - `za`, `zb`, `zc`, `zd`, … , `zy`, `zz`

Default Creation Mode

- `Blank` => `5.87`, `462.06`, `74`
- `Reals` => `5.87k`, `462.06m`, `74q`
- `Single Letters` => `5.87a`, `462.06g`, `74x`
- `Double Letters` => `5.87aa`, `462.06bd`, `74xy`

How to Reset Settings to Default Just press this button inside the settings panel

How to Use the API

Declaration

```
[Serializable]
public struct IdleCash : IEquatable<IdleCash>
```

Can be Used like other Unity struct types (Vector2, Vector3 etc.)

```
using UnityEngine;
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    public IdleCash money;
}
```

- Inspector View

Public Constructors

- `IdleCash(float value)`
- `IdleCash(float value, string type)`

```
using UnityEngine;
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    private void Start()
    {
        // Sets the value to 578.56 and Sets the type to first type
        IdleCash idleCash = new IdleCash(578.56f);
        // Sets the value to 578.56 and Sets the type to "bg"
        IdleCash idleCash = new IdleCash(578.56f, "bg");
    }
}
```

Public Static Properties

- `IdleCash Zero`
- `IdleCash One`
- `string FirstType`
- `string LastType`

```
using UnityEngine;
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    private void Start()
    {
        // Equivalent to 0
        IdleCash zero = IdleCash.Zero;
        // Equivalent to 1
```

```
        IdleCash one = IdleCash.One;

        // Default first type is ""
        string firstType = IdleCash.FirstType;
        // Default last type is "zz"
        string lastType = IdleCash.LastType;
    }
}
```

Public Properties

- IdleCash Simplified
- int TypeIndex
- float RealValue

```
using UnityEngine;
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    private void Start()
    {
        // Represents 1,000,000,000 or 1.00b
        IdleCash idleCashOne = IdleCash.One * 1_000_000_000f;
        // Simplified copy of "idleCashSecond"
        // Simplified means that its value field is between 1 and 1000
        // However, all IdleCash variables are most likely already simplified
        IdleCash simplified = idleCashOne.Simplified;

        // Represents 51.38 * 10^(3 * 6) or 51.38aa
        IdleCash idleCashTwo = new IdleCash(51.38f, "aa");
        // By default settings "aa" has index of 6
        int typeIndex = idleCashTwo.TypeIndex;

        // Represents 1,748,000,000,000 or 1.748t
        IdleCash idleCashThree = IdleCash.One * 1_748_000_000_000;
        // The real value of "idleCashThree" is 1,748,000,000,000 or 1.748E+12
        float realValue = idleCashThree.RealValue;
    }
}
```

Public Fields

- string type
- float value

```
using UnityEngine;
```

```
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    private void Start()
    {
        IdleCash idleCash = new IdleCash(578.56f, "bg");

        // Type of the "idleCash" is "bg"
        string type = idleCash.type;
        // Value of the "idleCash" is "578.56";
        float value = idleCash.value;
    }
}
```

Public Static Methods

- IdleCash Lerp(IdleCash a, IdleCash b, float t)
- IdleCash LerpUnclamped(IdleCash a, IdleCash b, float t)

```
using UnityEngine;
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    private void Start()
    {
        IdleCash a = new IdleCash(1, "k");
        IdleCash b = new IdleCash(1, "m");
        float t1 = 2.34f;
        float t2 = 0.55f;

        // Lerps between "a" to "b" depending on "t1"
        // "t1" is clamped between 0 and 1
        IdleCash.Lerp(a, b, t1);

        // Lerps between "a" to "b" depending on "t2"
        IdleCash.LerpUnclamped(a, b, t2);
    }
}
```

Public Methods

- bool Equals(IdleCash other)
- void Simplify()

```
using UnityEngine;
```

```
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    private void Start()
    {
        IdleCash idleCashOne = new IdleCash(15, "ao");
        IdleCash idleCashTwo = new IdleCash(15, "ao");

        // Returns true if both "idleCashOne" and "idleCashTwo" contains the same data
        bool equals = idleCashOne.Equals(idleCashTwo);

        IdleCash idleCashThree = new IdleCash(134.85f, "k");
        // Simplifies itself
        // However, all IdleCash variables are most likely already simplified
        idleCashThree.Simplify();
    }
}
```

Public Overridden Methods

- bool Equals(object other)
- int GetHashCode()
- string ToString()

```
using UnityEngine;
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    private void Start()
    {
        IdleCash idleCashOne = new IdleCash(15, "ao");
        object idleCashTwo = new IdleCash(15, "ao");
        // Returns true if both "idleCashTwo" is type of IdleCash and
        // "idleCashOne" and "idleCashTwo" contains the same data
        bool equals = idleCashOne.Equals(idleCashTwo);

        IdleCash idleCashThree = new IdleCash(49, "t");
        // Returns the Hash Code of the "idleCashThree"
        int hashCode = idleCashThree.GetHashCode();

        IdleCash idleCashFour = new IdleCash(134.85f, "k");
        // string form of "idleCash" is "134.85k"
        string toString = idleCashFour.ToString();
    }
```

```
}
```

Public Operators

- IdleCash +(IdleCash lhs, IdleCash rhs)
- IdleCash -(IdleCash lhs, IdleCash rhs)
- IdleCash *(IdleCash lhs, IdleCash rhs)
- IdleCash *(IdleCash lhs, float rhs)
- IdleCash /(IdleCash lhs, IdleCash rhs)
- IdleCash /(IdleCash lhs, float rhs)
- bool ==(IdleCash lhs, IdleCash rhs)
- bool !=(IdleCash lhs, IdleCash rhs)
- bool <(IdleCash lhs, IdleCash rhs)
- bool >(IdleCash lhs, IdleCash rhs)
- bool <=(IdleCash lhs, IdleCash rhs)
- bool >=(IdleCash lhs, IdleCash rhs)

```
using UnityEngine;
using EmreBeratKR.IdleCash;

public class Test : MonoBehaviour
{
    private void Start()
    {
        IdleCash idleCashLhs = new IdleCash(975.1f, "b");
        IdleCash idleCashRhs = new IdleCash(13.78f, "t");
        float floatRhs = 101.48f;

        IdleCash add = idleCashLhs + idleCashRhs;
        IdleCash substract = idleCashLhs - idleCashRhs;

        IdleCash multiply_IdleCash_IdleCash = idleCashLhs * idleCashRhs;
        IdleCash multiply_IdleCash_float = idleCashLhs * floatRhs;

        IdleCash divide_IdleCash_IdleCash = idleCashLhs / idleCashRhs;
        IdleCash divide_IdleCash_float = idleCashLhs / floatRhs;

        bool equals = idleCashLhs == idleCashRhs;
        bool notEquals = idleCashLhs != idleCashRhs;

        bool isSmaller = idleCashLhs < idleCashRhs;
        bool isGreater = idleCashLhs > idleCashRhs;

        bool isSmallerOrEqual = idleCashLhs <= idleCashRhs;
        bool isGreaterOrEqual = idleCashLhs >= idleCashRhs;
    }
```

```
}
```