Auction Calculator

Description: used to calculate taker amount and auction rate

Real world example

```
import { AuctionCalculator } from "@1inch/fusion-sdk";

const limitOrderStruct = {
  allowedSender: "0x0000000000000000000000000000000000000000",
  interactions:
    "0x000c004e2000000000000000000219ab540356cbb839cbe05303d7705faf486570009",
  maker: "0x00000000219ab540356cbb839cbe05303d7705fa",
  makerAsset: "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2",
  makingAmount: "1000000000000000000",
  offsets: "0",
  receiver: "0x0000000000000000000000000000000000000000",
  salt: "45118768841948961586167738353692277076075522015101619148498725069326976558864",
  takerAsset: "0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48",
  takingAmount: "1420000000",
};

const calculator = AuctionCalculator.fromLimitOrderV3Struct(limitOrderStruct);
// #=> AuctionCalculator instance

const rate = calculator.calcRateBump(1673548209);
// #=> 14285

const auctionTakingAmount = calculator.calcAuctionTakingAmount(
  "1420000000",
  rate,
);
// #=> '1422028470'
```

static AuctionCalculator.fromLimitOrderV3Struct

Description: used to create auction instance from limit order

Arguments: accepts LimitOrderV3Struct as an argument

Name	Type	Inner Solidity Type	Description
salt	string	uint256	some unique value. It is necessary to be able to create limit orders with the same parameters (so that they have a different hash)
makerAsset	string	address	the address of the asset user want to sell (address of a token contract)
takerAsset	string	address	the address of the asset user want to buy (address of a token contract)
maker	string	address	the address of the limit order creator
receiver	string	address	If it contains a zero address, which means that taker asset will be sent to the address of the creator of the limit order. If user set any other value, then taker asset will be sent to the specified address
allowedSender	string	address	If it contains a zero address, which means that a limit order is available for everyone to fill. If user set any other value, then the limit order will be available for execution only for the specified address (private limit order)
makingAmount	string	uint256	amount of maker asset
takingAmount	string	uint256	amount of taker asset
offsets	string	uint256	every 32's bytes represents offset of the n'ths interaction
interactions	string	bytes	used to encode fusion specific data

Order.interactions suffix structure:

- M*(1 + 3 bytes) - auction points coefficients with seconds delays
- N*(4 + 20 bytes) - resolver with corresponding time limit
- 4 bytes - public time limit (started from this point of time an order can be full filled by anyone)
- 32 bytes - taking fee (optional if flags has _HAS_TAKING_FEE_FLAG)
- 1 byte - flags

Examples:

```
import { AuctionCalculator } from "@1inch/fusion-sdk";

const limitOrderStruct = {
  allowedSender: "0x0000000000000000000000000000000000000000",
  interactions:
    "0x000c004e2000000000000000000219ab540356cbb839cbe05303d7705faf486570009",
  maker: "0x00000000219ab540356cbb839cbe05303d7705fa",
  makerAsset: "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2",
  makingAmount: "1000000000000000000",
  offsets: "0",
  receiver: "0x0000000000000000000000000000000000000000",
  salt: "45118768841948961586167738353692277076075522015101619148498725069326976558864",
  takerAsset: "0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48",
  takingAmount: "1420000000",
};
```

```
AuctionCalculator.fromLimitOrderV3Struct(limitOrderStruct);
// #=> AuctionCalculator instance
```

AuctionCalculator.calcRateBump

Description: used to calculate exchange rate in some point of time. user can read more about it here

Arguments: time (unix timestamp)

AuctionCalculator.calcAuctionTakingAmount

Description: used to calculate taker amount

Arguments:

- [0] takingAmount: string
- [1] rate: number

AuctionCalculator.fromAuctionData

Description: creates AuctionCalculator from suffix and salt

Arguments:

- [0] suffix: AuctionSuffix
- [1] salt: AuctionSalt

Example:

```
import {
  AuctionSuffix,
  AuctionSalt,
  AuctionCalculator,
} from "@1inch/fusion-sdk";

const suffix = AuctionSuffix.decode(
  "0x000c004e2000000000000000000219ab540356cbb839cbe05303d7705faf486570009",
);
const salt = AuctionSalt.decode(
  "45118768841948961586167738353692277076075522015101619148498725069326976558864",
);
AuctionCalculator.fromAuctionData(suffix, salt);
// #=> AuctionCalculator instance
```