

t get turbent state (): ant † get turbent state (): ant † advance state (degrical): with † advance state (degrical): with † get turbent state (degrical): with † get turbent state (degrical): with † get turbent state (limit † get turbent † get turbent state (limit † get turbent † get t	stou	stoupront
t add thint [key; thing; clint]: noted t add thint [key; thing; clint]: noted t get threated & scheme (): int t get thint scheme (): int t get thint scheme (): int t add thint [key; thing; thing): (ellution (sale) t key for Stocked historication (court: thing gradene): "Tring): bottom t add thint [key; thing; mann; string] allow; tring]: note t add Stocked [key; thing; mann; thing]: note t get Product [key; thing]: schemet t demand string the scheme (key; thing): schemet t demand string the scheme (key; thing): schemet t demand string the scheme (key; thing): schemet t demand string (key; thing); schemet t add scopelier [cey; thing]: schemet t get tryphin scheme attern (key; thing): string): scheme t get tryphin scheme attern (key; thing): string): bottom t get tryphin scheme attern (key; thing): string): bottom t get tryphin scheme attern (key; thing): string): bottom t get scheme attern (key; thing): string): scheme t get scheme attern (key; thing): scheme t get scheme attern (key; thing): scheme t get scheme attern (key; thing): scheme t looked scheme attern (key; thing): scheme t get thing scheme t	- accounting Dolance: int = 0	bilmand: - Itring = ""
t activate All Frederics (c) (Whine) i wild t add Sodaut (ky: Wing): Gradut i wild t add Sodaut (ky: Wing): Gradut i wild t get All dient (ky: Wing): delection (ky: wild): wild t get And Experience (ky: Wing): graduat i wild t day from (ky: Wing): graduat i wild t add Experience (ky: Wing):	t add Wint (key: Wing, c: Wint): void t get Wint (key: Leving): Wint t get Wint (key: Leving): Wint t get Wint Gransactions (key: String): Collection (Laly)	+ get truvent State (): int + get Avoilable Galance (): int + get Accounting Galance (): int + add Wint (key: Wing, name: String, address: string): void
t add Supplier (ky: Wing): Supplier ; tring (): Supplier ; tring (): String	+ old Stoduct (key: thing, p: Broduct): void + get Broduct (key: thing): Broduct	+ get All blints (): tolliction (blint) + get blint (key; string): blint + get blint transactions (key; string): welletion (sale)
todd scansaction (key; int): Stromsaction): rood + get Scansaction (key; int): Stromsaction + get Scansaction (key; int): Stromsaction + pay (key: int): roid + pay (key: int): roid + get All Scaleta (): cellution & Scaleta + lookup Scaleta under Sop Scale (limit; int): tellution & Scaleta + get All Scaleta (product: thring); roid import side (extfile: thring): roid + get All suppliers (): tellution & Scaleta + get All suppliers (): tellution & Suppliers + get Import Scansactions (key: thring): tellution & Corden 7 + get Import Scansactions (key: thring): tellution & Corden 7 + toggle Scansactions (key: thring): believe + add tale (limit: thring): roid + add tale (limit: thring): roid + get Scansaction (key: int): roid + get Scansaction (key: int): Scansaction + pay (key: int): roid	+ get supplier (key: String); supplier + get supplier (key: String); supplier + get supplier Stansactions (key: String); tellection (brown)	iston: String, price: int, critical later; int, supplier: string); world
+ looking Stoderells Mindel Sop Stree (limit : int): world in port Sile (tetfile: Thring): world + get the supplier Strong consections (key: thring): tellustion (when to toggle Transactions (key: thring): tellustion (when toggle Transactions (key: thring): bellustion (when toggle Transactions): supplier: thring): best with the comment in the products: that he comment is world + add tale (limit: thring, product: thring, and the int, product: thring, and the consection (key: int): world + get Strongaction (key: int): Transaction + pay (key: int): world	+ get Gransaction (key: int): Gransaction): void + get Gransaction (key: int): Gransaction + pay (key: int): void	+ and trentainer (key; Itring , price; int, critical balue; int supplied: Itring, price; int, critical balue; int supplied: Itring, eyel: string, level: tring): a
toggle Fransactions [supplier: Ithing]: broken + add water (supplier: Itring, products: Hash rap (storing, int: : rooid + add tale (dint: Itring, deadline: int, product: terring, omount: int): rooid + get Fransaction (key: int): Fransaction + pay (key: int): rooid	+ lookup Graduates under Son arter (limit: int): Wellection (Graduet)	+ change Strice (product: String, price: int): wid + add supplier (key: string, mome: string, address: string): w + get All suppliers 1): tellection (suppliers
t get From saction (Key; int): From saction + pay (Key; int): void		+ add water (supplier: string); broken + add water (supplier: string, products: Hash hap (storing, int)
		+ get Stomsaction (Key; int): Stomsaction





