Keelan Matthews – Practical 1

u21549967

Task 1

1.1 a – stack – it is created **at** run time
   b – stack – the pointer itself is created on the stack, the memory it is pointing to is allocated on the heap.
   c[10] – stack – it is created at runtime with a fixed size
   d – stack – it is created at runtime with a fixed size
   e – heap - it is assigned memory **during** runtime (pointer)
   f – heap - it is assigned memory **during** runtime (pointer)
   g – stack – it is created at runtime
   h – stack – it is created at runtime
   n – stack – it is created at runtime

1.2 It would not work as h is a constant variable and cannot be changed later. Assigning it NULL renders it useless.

1.3 - g is a char type variable, but is being assigned an integer value
   - e pointer is being pointed to a value rather than an address in memory
   - c array is not a pointer array, but it is being assigned an address

Task 2

2.1 Before the constructor of the derived class.

2.2 After the destructor of the derived class.

2.3 After

2.4 Class A -> Class B -> Class D

2.5 Class D -> Class B -> Class A

Task 3

3.1 See code

3.2 This worked because division was performed between two integer types.

3.3 This worked because addition was performed between two floating point types.

3.4 This did work because strings can be concatenated without overloading the operator.

3.5 This did not work because it is not possible to multiply string data types without overloading the multiplication operator.

Task 4

4.1 “15 15” – ptr_a is dereferenced and assigned the value 15, and ptr_b points to the same address of ptr_a. They are both dereferenced in the cout.

“15 4” – ptr_b is now made to point to an address that stores 4.

“15 15” – ptr_b now points back to the same address as ptr_a

“15 15” – ptr_a is deleted and made to point to the same address as ptr_b, which is still pointing to the address that ptr_a was pointing to before deletion. Each “&*” pair cancels out which leaves ptr_b being dereferenced.

“*address of ptr_a* 15” – ptr_c points to the address of ptr_a. When dereferenced it returns the address of the pointer. When it is dereferenced twice it returns the value that ptr_a is pointing to.

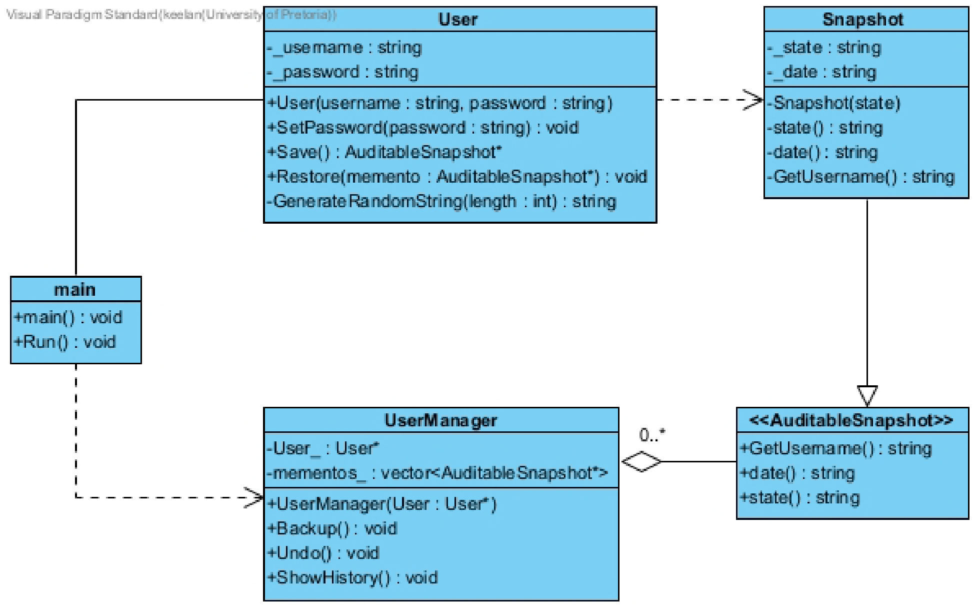
Task 6

6.1 AuditableSnapshot

6.2 Snapshot

6.3 User

6.4 UserManager


Visual Paradigm Standard(keelan(University of Pretoria))
User
-_username : string
-_password : string
+User(username : string, password : string)
+SetPassword(password : string) : void
+Save() : AuditableSnapshot*
+Restore(memento : AuditableSnapshot*) : void
-GenerateRandomString(length : int) : string
Snapshot
-_state : string
-_date : string
-Snapshot(state)
-state() : string
-date() : string
-GetUsername() : string
main
+main() : void
+Run() : void
UserManager
-User_ : User*
-mementos_ : vector<AuditableSnapshot*>
+UserManager(User : User*)
+Backup() : void
+Undo() : void
+ShowHistory() : void
0..*
<<AuditableSnapshot>>
+GetUsername() : string
+date() : string
+state() : string

6.5