IN3020/IN4020 – Database Systems
Spring 2021, Week 10.1

DBMS Architecture:
Components & Interdependencies

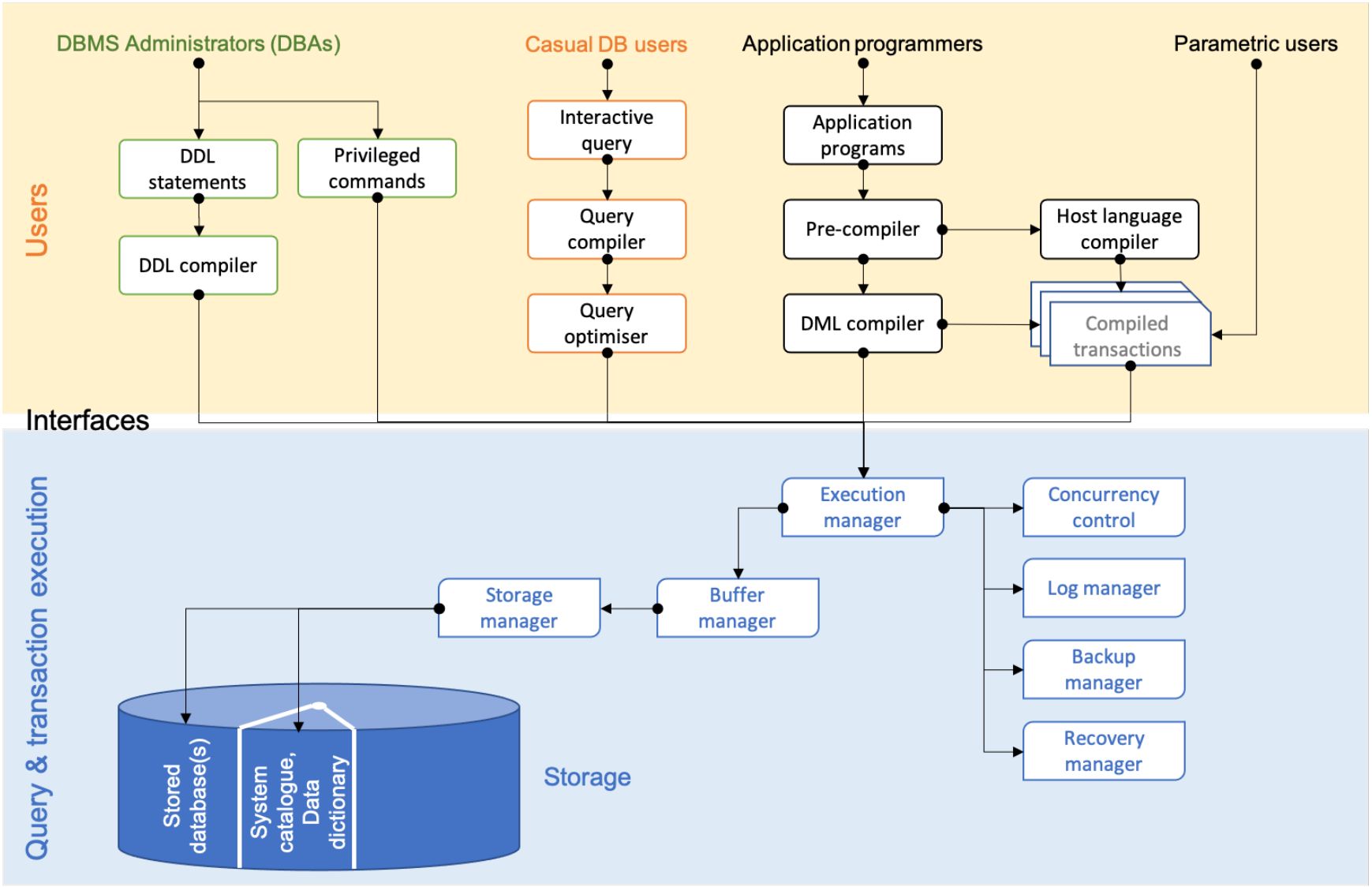
FOR IN4020
recommended but not required for IN3020

Dr. M. Naci Akkøk

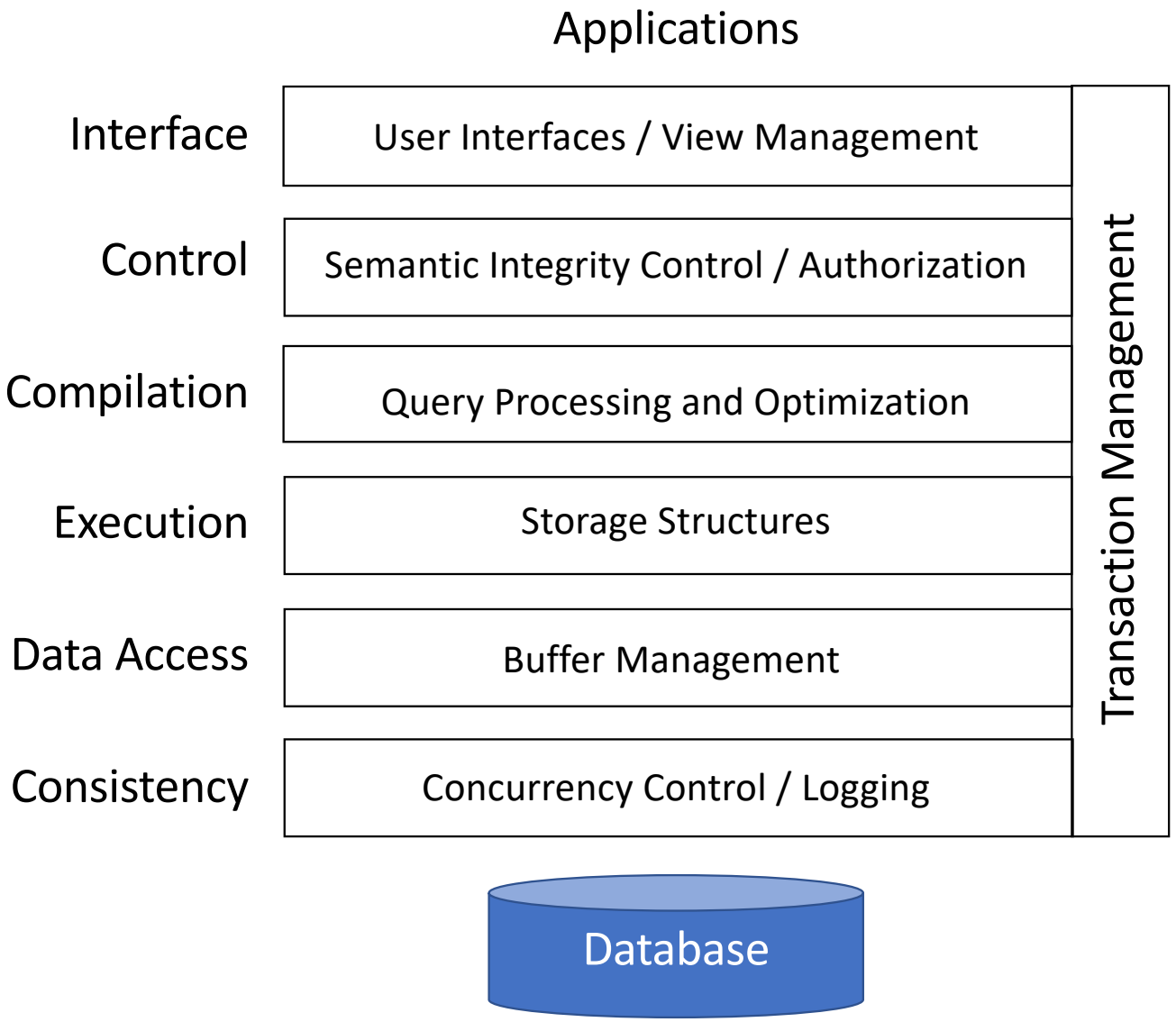
CEO, In-Virtualis, Assoc. Prof. UiO/Ifi, Assoc. Prof. OsloMet/CEET

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Example DBMS architecture and its components



Same architecture as functional layers



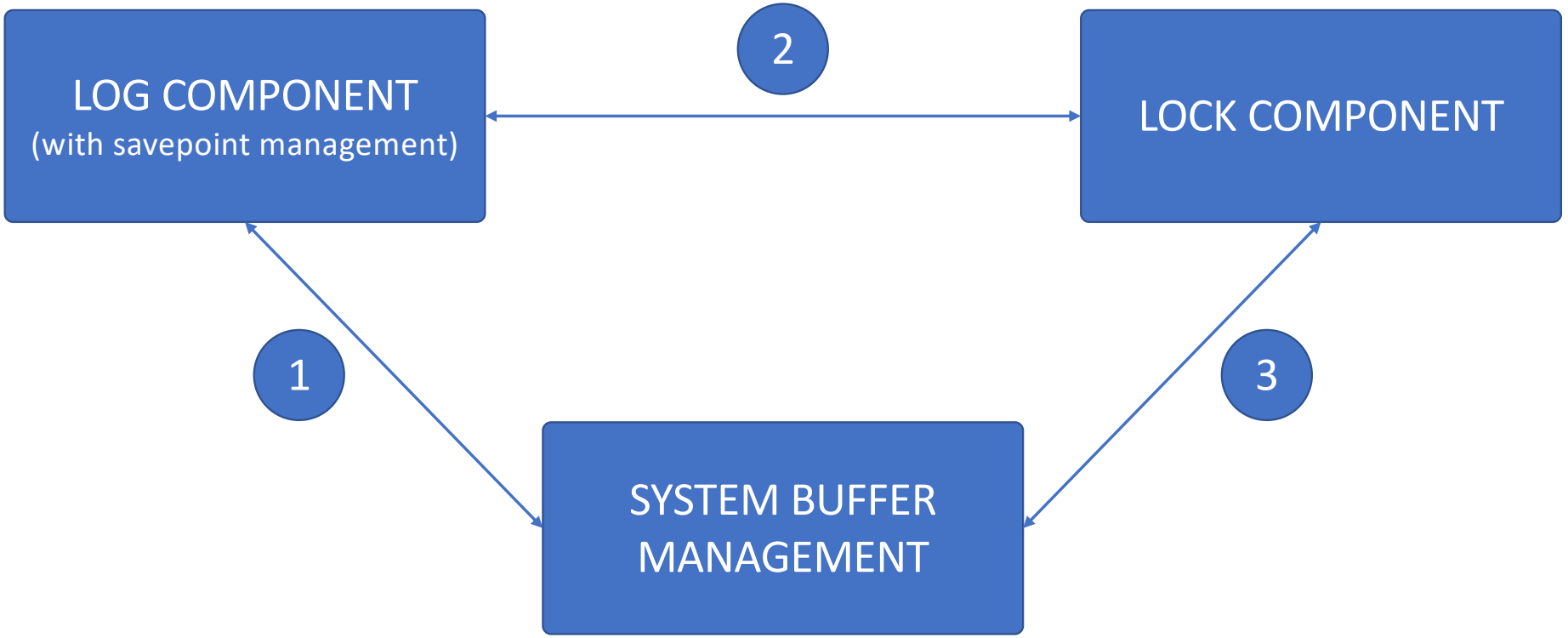
Dependencies – Example

- Understanding dependencies between these components is important!
- Remember “Group Commit”?
  - With group commit, locks can be released earlier than with the strict locking rule
  - Group commit: Transaction or Concurrency Manager | Lock Manager
    - A **transaction** cannot release any write **lock** until it has committed or aborted, and the commit or abort log entry is written to (primary) **memory** Buffer Manager
    - **Log blocks** must be flushed to the log disk in the order they are in the primary memory Log Manager

Dependencies between layers/components

- **Central components**: components that manage their resources directly down to the operating system interface - system buffer management, lock component, log component (with savepoint management).
- **Higher-level components**: components demanding/requiring the central components as prerequisites - transaction management, access path management, sorting component, etc.

Dependencies between central components



Interaction 1: Log & Buffer (pool)

- WAL (write ahead log) principle: log information is written to secondary storage before the “real” information (pages) is written to secondary storage.
- Page replacement strategy determines which protocol (logging) approaches are applicable. For instance, logical protocol approach rquires indirect replacement strategy.

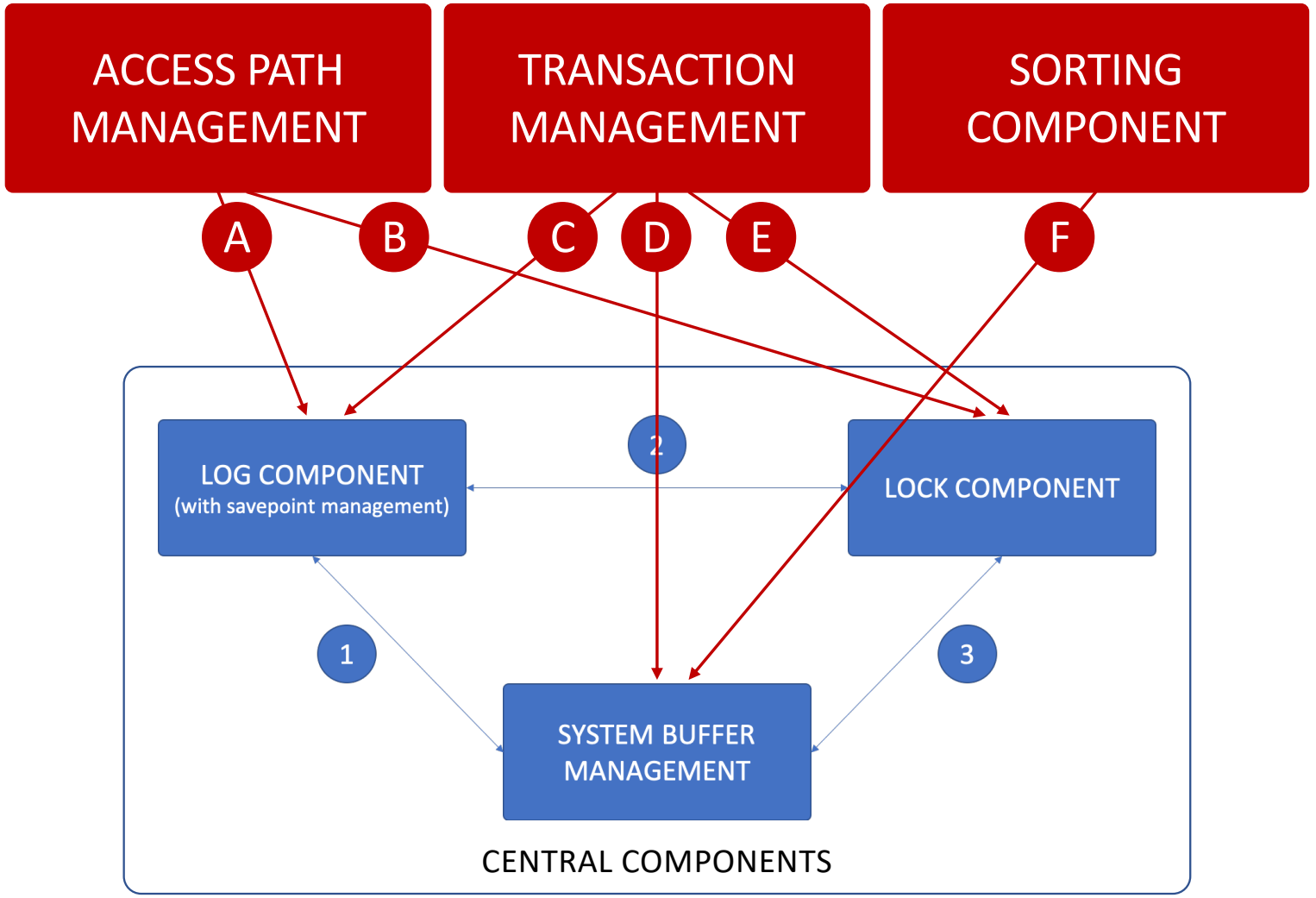
Interaction 2: Log & Lock

- Logging unit ≤ Locking unit
- In case of logical protocol approaches, the logging unit is the set of all data objects changed by a DML operation. If a rollback is performed no other transactions must be affected (damaged).
- If the lock component realizes only a simple 2-phase-locking protocol - instead of a strict 2-phase-commit protocol (all locks are kept until end-of-transaction), all relationships (interferences) between transactions (also read-only) about commonly used data items must be logged to enable a recursive rollback in case of failure.

Interaction 3: Lock & Buffer (pool)

- The main task of the lock component is to guarantee the “logical single user mode”.
- This requires that the lock component controlls which pages a fixed and which pages are unfixed in the system buffer by the system buffer management (influencing paging strategy) to guarantee isolation (locking).

Dependencies between central & higher-level components



Interaction A: Log & Access path mgmnt.

- Physical state logging:
  Requires the maintenance and logging of access paths when inserting new data items.
- Logical state logging: requires no extra effort of access path maintenance because DML operations can be rolled back (inverted) and repeated.

Interaction B: Lock & Access path mgmnt.

- For B* trees (special access paths) an unadjusted locking concept can decrease parallelism. No exclusive locks should be set on the root of the tree, because this would block all access paths along that tree.

Interaction C: Transaction mgmnt. & Log

- The “all-or-nothing” principle of a transaction requires the logging of UNDO information for rollback and REDO information to repeat a transaction.
- In case of very many short transactions the logging component can become the bottleneck of the DBMS.
- One solution to this problem is to group transactions and to defer their end-of-transaction in order to write the log information in a blocked way to secondary storage.

Interaction D: Tx & System buffer mgmnt

- The system buffer management is the central component concerning performance optimization.
- The number of parallel transactions has a strong influence on the paging rate.
- Transaction management should defer the activation of a transaction if this would cause a decrease of the paging rate.
- Transaction management should group transactions in a buffer-oriented way for performance reasons.

Interaction E: Tx management & Lock

- The locking component guarantees the transaction-oriented isolation of data items.
- It is possible that deadlock situations (circular resource allocations) occur that require “unjustified” rollbacks of transactions to resolve the deadlock.
- Transaction management has to determine which transaction(s) have to be rolled back in order to cause minimum work loss.
- Transaction management and locking component have to be adjusted so that deadlocks are not possible, e.g., by pre-claiming(*), or sequentialization of transactions which have overlapping data areas.

Pre-Claiming Lock Protocol

- Pre-claiming protocols evaluate their operations and create a list of data items on which they need locks in advance.
- Before initiating an execution, the transaction requests the system for all the locks it needs beforehand.
  - **If all the locks are granted**, the transaction executes and releases all the locks when all its operations are over.
  - **If all the locks are <u>not</u> granted**, the transaction rolls back and waits until all the locks are granted.

Interaction F: Sorting component & System buffer management

- Special precautions for system buffer management are necessary if the sorting component is active (especially relevant for relational DBMS).
- The paging strategy should not be contra-productive to the applied sorting algorithm.

SEE YOU IN ABOUT 2 WEEKS!
(Monday 12th April)

STAY SAFE, STAY HEALTHY!