A variable's data type (or simply type) determines its possible values and operations.

```java
int course;
course = 33;
course = -33;
course = 3.14;

String course;
course = "33";
course = "-33";
course = "3.14";

(33 + 2) == 35
("33" + "2").equals("332")

course.equals(33) Can't call equals
```

?: What is an example of an impossible value and an impossible operation for String?

?: Why not just use ArrayList all the time?

Java interfaces represent the software design concept of abstract data types. An abstract data type is a data type that does not specify any one implementation. Data structures implement ADTs.

List ADT A collection storing an ordered sequence of elements.
- Each element is accessible by a zero-based index.
- A list has a size defined as the number of elements in the list.
- Elements can be added to the front, back, or any index in the list.
- Optionally, elements can be removed.

Resizable array can implement List, Stack, Queue, Deque, PQ, etc.
Linked nodes can implement List, Stack, Queue, Deque, PQ, etc.
Abstract data types hide implementation details from clients (users of ADTs).

**Contract**
Description of an ADT's possible values and operations

What are the consequences of breaking the contract?
When might it be useful to know the implementation details of an ADT's values or operations?

**Q1:** What are the consequences of breaking the contract?

**Q2:** When might it be useful to know the implementation details of an ADT's values or operations?

In an ArrayList, the i-th item of the list (ADT) is always stored at array index i (data structure). **Give an example of an operation that is very slow on ArrayList.**

**Q1:** Give an example of an operation that is very slow on ArrayList.