

CISC124 – Today's Topics

- Arrays of objects
- Encapsulation (data & behaviour)
- Designing methods
- Functional decomposition
- Recursion

Winter 2019 F. de la Parra 1

•1

Arrays of objects

- Arrays of primitive types -> int[], float[], char[], boolean[]
 - Array size is fixed and all stored values are of the same primitive data
 - Declare -> int[] digits = new int[10]; and
 - Initialize -> for (int i=0; i < 10; i++) {digits[i] = i;}
 - Or declare-initialize -> int[] digits = {0,1,2,3,4,5,6,7,8,9};
- Arrays of class types -> any user-defined class or Java API.
 - Array size is fixed and all stored values are of the same class type
 - Declare -> Student[] cList = new Student[200]; and
 - Initialize -> cList[0] = new Student("John", "Smith", 1);
cList[1] = new Student("David", "Jones", 2);
etc.

Winter 2019 F. de la Parra 2

•2

Encapsulation

- Code in a class encapsulates
 - Data
 - Attributes are private and keep track of an object's state
 - Their values get accessed or changed by using methods (accessor and mutator methods)
 - Behaviour
 - Implemented in methods other than accessors and mutators
 - Methods use the state values stores in attributes and return values to method invocations

Winter 2019 F. de la Parra 3

•3

Encapsulation

```

Public class myClass {
    // 1) Class variable declaration area
    // 2) Constructor methods area
    // 3) Mutator methods area
    // 4) Accessor methods area
    // 5) Object-behaviour methods area
}
    
```

Winter 2019 F. de la Parra 4

•4

Designing methods

Winter 2019 F. de la Parra 5

•5

Functional decomposition

- Task is solved by one (or more) collection of objects having a specified role in its completion
- Roles 1, ... n in the collection of objects should be similar
- Although their states could be different, the functions they need to perform to fulfill their roles are identical.
- A function performed by an object can be split up into multiple functions
- This functional decomposition can continue to many levels down a hierarchical tree

Winter 2019 F. de la Parra 6

•6

Recursion

- Function performs an initial task
- Task can be decomposed into smaller subtasks of the same nature as the initial task
- Smaller subtasks can be decomposed into even smaller subtasks of the same nature as the previous subtasks
- This recursive subtask splitting continues until subtask decomposition is not possible any more
- In terms of a java code, this translates to a method being able to call itself until a point where no more invocations of itself are possible

Winter 2019

F. de la Parra

7

•7