CS211 Section 2

Scanner
Induction
Recursion

Nomenclature

- Recitation
  - Means the recitation you signed up for..
  - E.g. {1, 2, 3, 4, 5, 6, 7, 8}
- Section
  - Each successive section
  - Week after Week…
- Section Website!
  - Will post these slides as PDF later today!

Scanner

- Constructors:
  - Scanner s = new Scanner(*String*)
  - Scanner s = new Scanner(*File*)
  - Scanner s = new Scanner(*InputStream*)
  - Scanner s = new Scanner(*Readable*)
    - Scanner s = new Scanner(*InputStreamReader*)
    - Scanner s = new Scanner(*BufferedReader*)
    - *Etc…*
- *Nice Thing: Don't need try…catch*

Scanner *scanning*

- Some Important Methods:
  - Scanner s = new Scanner(…);
  - s.useDelimiter(String delim); //i.e. delim = " .,!"
  - s.hasNext(), s.next()
  - s.hasNextInt(), s.nextInt()
  - Etc.. You can check the api for all the other methods…
- Kind of like an ***iterator*** for the *tokens* in a string separated by *delimiters…*

Parsing

- How to parse any input source?
  - First, specify *delimiters…*
    - s.useDelimiter(String delim)
  - Next, check to see if there are more tokens…
    - s.hasNext()
  - Then, get/eatup the next token…
- Any questions…?

Console Input/Output

- `Scanner s = new Scanner(System.in);`
- `System.out.println("CS211 Rocks!");`
- `System.err.print("CS211 ☹\n");`

File *Input*

- Pass a *File* object into the constructor of the Scanner…
- Treat the same as any other scanner..
- For multi-line files, two options:
  - Make **"\n\r"** part of the delim string
  - Use **hasNextLine()** and **nextLine()** to traverse to the next line.
- ***ALWAYS*** remember to close the Scanner at the end:
  - s.close()

File Output

- Need java.io.*
- Need try{…}catch(){…} block to catch IOException
- Construction:
  - `PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("foo.out")));`
- To Write:
  - `out.print(…);`
  - `out.println(…);`
- ***ALWAYS*** remember to close the stream at the end: **out.close()**

Mathematical Induction

☺

INDUCTION, **STRONG**

- To prove a property P(n) for all natural numbers, do this:
  – (1) Prove the base case: P(0)
  – (2) Prove the inductive case:
    - Inductive Hypothesis: For arbitrary positive integer k: assume that P(0), P(1), … , P(k-1) are true.
    - Prove P(k).

STRONG INDUCTION EXAMPLE

- Every integer >= 2 is divisible by a prime
- Theorem. P(n) holds, for n >= 2, where P(n): Every integer n >= 2 is divisible by a prime. (A prime is an integer >= 2 whose only positive divisors are 1 and itself)
- Proof.
  – Base case: P(2) is true because 2 is a prime and divides itself.

STRONG INDUCTION EXAMPLE

- Inductive case: We assume the Inductive hypotheses P(2), P(2), ..., P(k): that is, we assume that each of these is divisible by some prime, say p.
- Consider integer (k+1). We show that P(k+1) holds by case analysis:
  – If k+1 is a prime, it is divisible by itself, so P(k+1) holds.
  – If k+1 is not a prime, then (k+1) = a*b where 2 <= a <= k+1 and 2 <= b < = k+1. By induction hypothesis P(b), we know that b is divisible by a prime.
- Since b divides k+1, k+1 is also divisible by a prime, so P(k+1) holds.

INDUCTION, *WEAK*

- To prove a property P(n) for all natural numbers, do this:
  - (1) Prove the base case: P(0)
  - (2) Prove the inductive case:
    - <u>Inductive Hypothesis:</u> For arbitrary positive integer k: assume that P(k-1) is true
    - Prove P(k).

WEAK INDUCTION EXAMPLE

- Prove by induction that any amount greater than 14 can be obtained using 3-cent and 8-cent coins.
- Theorem. For all n, n⩾ 14, P(n) holds P(n): Some bag of 3-cent and 8-cent coins has sum n.
  - Base case P(14). A bag with two 3-cent coins and one 8-cent coins sums to 14.

WEAK INDUCTION EXAMPLE

- Inductive case. Assume Inductive hypothesis P(n) and prove P(n+1). Since P(k) holds, there is a bag of 3-cent and 8-cent coins that sums to k. Consider two cases: the bag contains an 8-cent coin or it does not.
  - Case 1: the bag contains an 8-cent coin. Take the 8-cent coin out and put in 3 3-cent coins. The bag now sums to k+1. Case proved.
  - Case 2: the bag doesn't contain an 8-cent coin. The bag contains only 3-cent coins. Since k>=14, the bag contains at least five 3-cent coins. Take five 3-cent coins out and throw in two 8-cent coins. The bag now sums to k+1. Case proved.

HORSES! HORSES! HORSES!

- Theorem: All horses have the same color!
- Theorem: For all n>=1, P holds:
  - P(n): all horses in a group of n horses have the same color.
- Proof.
  - Base Case. P(1) holds, obviously because every horse in a group of 1 has the same color.

HORSES! HORSES!

- Inductive case. Assume inductive hypothesis P(n) and prove P(n+1):
  - Consider a group of n+1 horses. Expose P(n) by singling out 1 horse, say H1, and removing it from the group. We now have a group of n horses and by inductive hypothesis P(n), they all have the same color.
  - Put H1 back into the group and pull out another horse, H2, giving us another group of size n. By P(n), they all have the same color, and since H1 is in the group, they all our colored them same as H1. But by the first paragraph above, they all have the same color as H2.
- Therefore, all n+1 horses have the same color, and P(n+1) is proved. What is wrong with the proof?

HORSES!

- P(2) is not true and cannot be proved. The proof in the inductive case is faulty. The group of two consists of {H1, H2}.

- Go over the inductive case carefully in the case n=1 and you are trying to prove P(n+1), or P(2), and you will see the falacy.

Time to try it on your own… Or w/ Neighbor

- Theorem. For all n >= 0, P(n) holds, where
  - P(n): sum of 0..n = n(n+1)/2

The Answer…

- Base case: Prove P(0). The sum of 0..0 is 0, and 0(0+1)/2 is also 0.
- Inductive case: Assume inductive hypothesis P(n) and prove
  - P(n+1): P(n+1): sum of 0..n+1 = (n+1)(n+1+1)/2
  - We start with the lefthand side of P(n+1) and transform it into the right side

The Answer… *continued*

- sum of 0..n+1
- = <Split off last term, to expose P(n)>
- sum of 0..n + (n+1)
- = <Use P(k)> n(n+1)/2 + (n+1)
- = <arithmetic> n^2/2 + n/2 + n + 1
- = <arithmetic> (n^2 + 3n + 2)/2
- = <arithmetic> (n+1)(n+2)/2

- **P(n+1): sum of 0..n+1 = (n+1)(n+1+1)/2**

Another example …

- Prove by induction that:
  - for n>=3: 2n+1 < 2^n.

Answer…

- Base Case: n = 3: 2(3) +1 < 2^3

  7 < 8
- Inductive Case: Assume Inductive hypothesis P(k), where k ⩾ 3; prove P(k+1):
  - P(k+1): 2(k+1)+1 < 2^(k+1)

- 2(k+1) + 1 ?< 2^(k+1)
- <Expose IH> (2k+1) + 2 ?< 2(2^k)
- <Algebra: RHS> (2k+1) + 2 ?< 2^k + 2^k
- **IH:** We know that 2k+1 is < 2^k, hence, we r left with:
- [<2^k] + 2 ?< 2^k + 2^k
- Since 2 < 2^k for any k >= 3, we can conclude that, in fact:
- (2k+1) + 2 < 2(2^k)

Recursion

Adapted from Rouzzi & Minich

What is Recursion?

- Induction in REVERSE
- Start with *K*th state.
- Reduce it to *K-1*th state.
- Continue until BASE case reached…
- Recursive functions should have
  - base case(s) – usually a trivial case
  - recursive step – breaks the problem up into smaller problems that are recursive

Simple Example

- Iterative Sum:
  - int sum =0;
  - for(int i=1; i< =k; i++)
  - sum += i;
- Recursively:
- public int sum(int n) {
  - if(n==0) //base case
    - return 0;
  - else //recursive case
    - return sum(n-1) + n;
  - }

Where the nth state is added to the (n-1) th state for all n greater than the base case. Which in this example was 0.

Tail Recursion

- A method is tail recursive if the last action of the recursive method is the recursive call.

Example

- **Compute n mod m without using %:**
- public int modulus(int val, int divisor) {
  - if(val < divisor)
    - return val;
  - else
    - return modulus(val - divisor, divisor);
- }