1.00 Tutorial 11

Trees

Streams

pset10

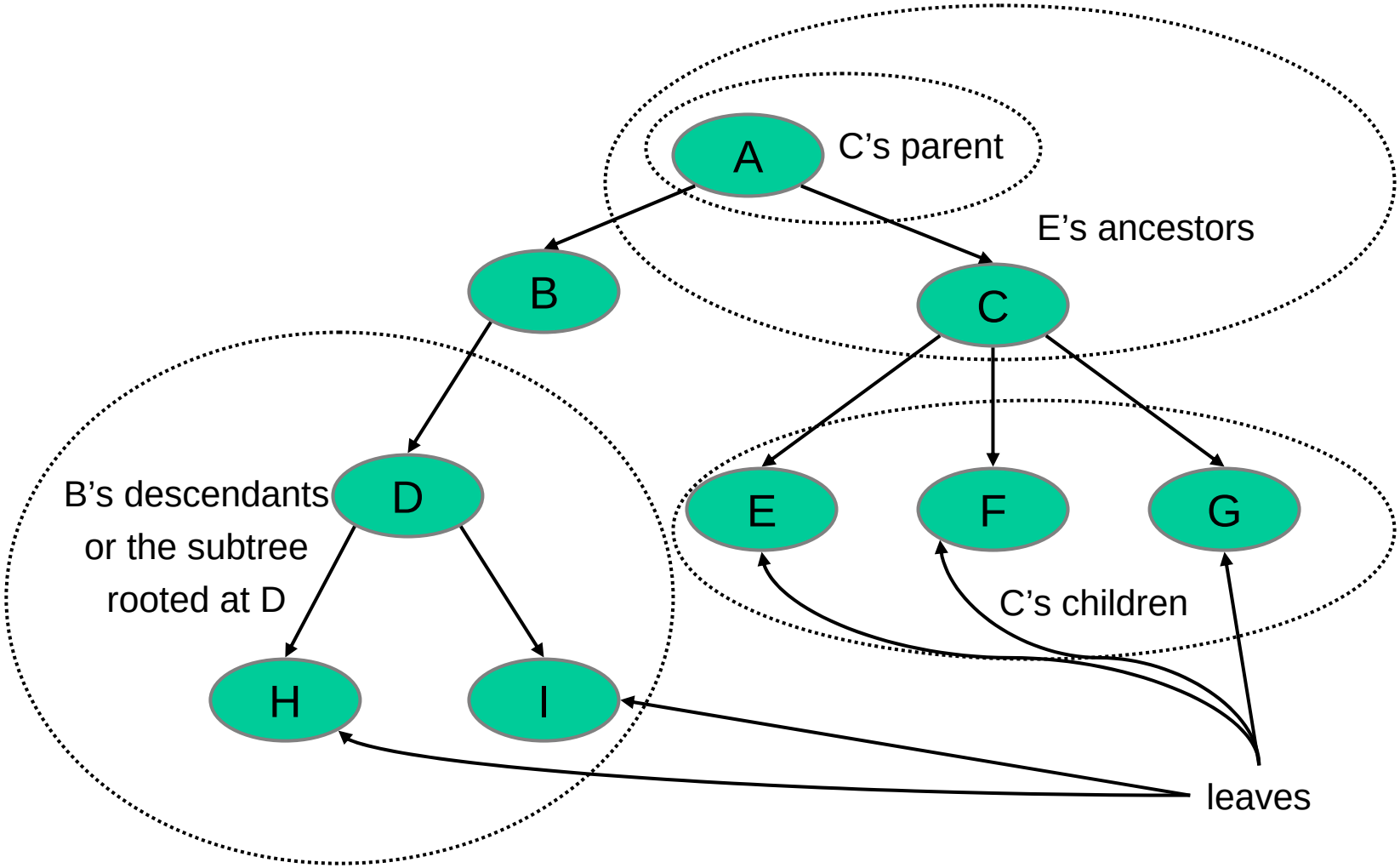
Outline

- Administrative stuff
- Trees
- Streams
- PS #10

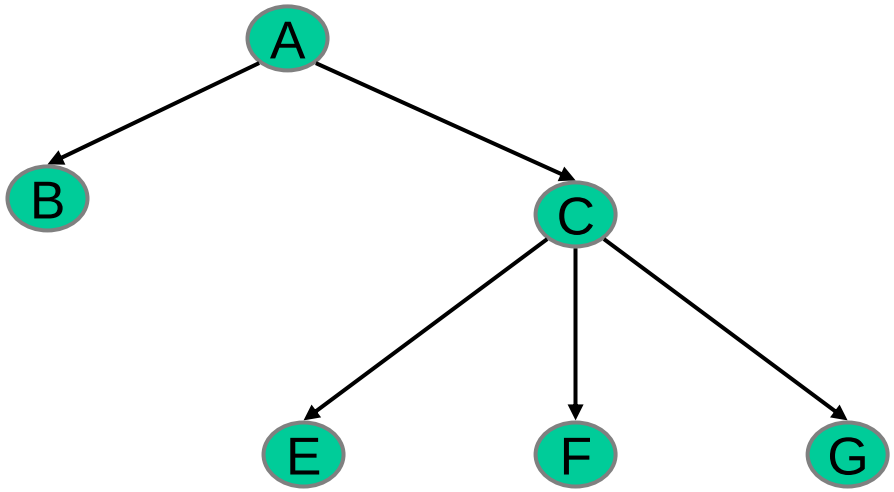
Administrative Stuff

- Check your grades on MIT server and with TA before final exam
- Office Hours:
  - This week – normal
  - Next week - Review Session and May 12
- Final
  - Covers lectures 1 to 36
  - Wed May 18, 1:30pm-4:30pm,
  - Final Review: Wednesday, May 11th, 7-9PM
  - Open book, open notes
- Return laptops @ final exam, or beforehand (MWF 10-2). You must bring all your accessories too. (Duh.)

Tree Terminology



Implement this tree



- Node class
  - Needs to hold arbitrary number of children
- Tree class
  - Implement this tree in buildExampleTree

Streams Overview

- Java programs communicate with the outside world using streams
  - Good way to input lots of info.
  - Good way to store output and communicate results to other programs, perhaps written in other languages entirely
- I/O streams: work in one direction only
  - Input stream: control data coming into the program
  - Output stream: control data leaving the program
- Streams: FIFO queues
  - First In First Out

General Strategy for reading from and writing to a Stream

- Reading
  - Open a stream
  - while more information
    - read information
  - close the stream
- Writing
  - Open a stream
  - while more information
    - write information
  - close the stream

Important abstract Stream classes

- InputStream
  - Read bytes
- OutputStream
  - Write bytes
- Reader
  - Read chars
- FileWriter
  - Write chars

Important Stream Classes

- FileInputStream
  - Read data in binary format from files
- FileOutputStream
  - Write data in binary format to files
- FileReader
  - Read text data from files
- FileWriter
  - Write text data to files

Connecting Streams

- Each stream class has a specific functionality.
- Streams can be connected to get more functionality

- Example
  `BufferedReader`
  Buffers the character stream from `FileReader` for efficiency and allows you to read line by line

  FileReader input = new FileReader("C:\\test.txt");
  BufferedReader bufferedIn = new BufferedReader(input);

- Use StringTokenizer to break a string into smaller pieces, or *tokens*:

```
StringTokenizer toke=newStringTokenizer(str,”\t”); //split on tab
String thisToken=toke.nextToken();
```

Example

```
try
{
   Reader reader = new FileReader(“input.txt”);
   int next = reader.read();
   char c;
   if (next != -1){ //check for end of input
       c = (char)next;
       // do something with the character c
   }
   char[] buf = new char[512];
   int nRead = reader.read(buf);
   if(nRead != -1)
       { // do something with the char array buf }
}
Catch (IOException e)
{ }
```

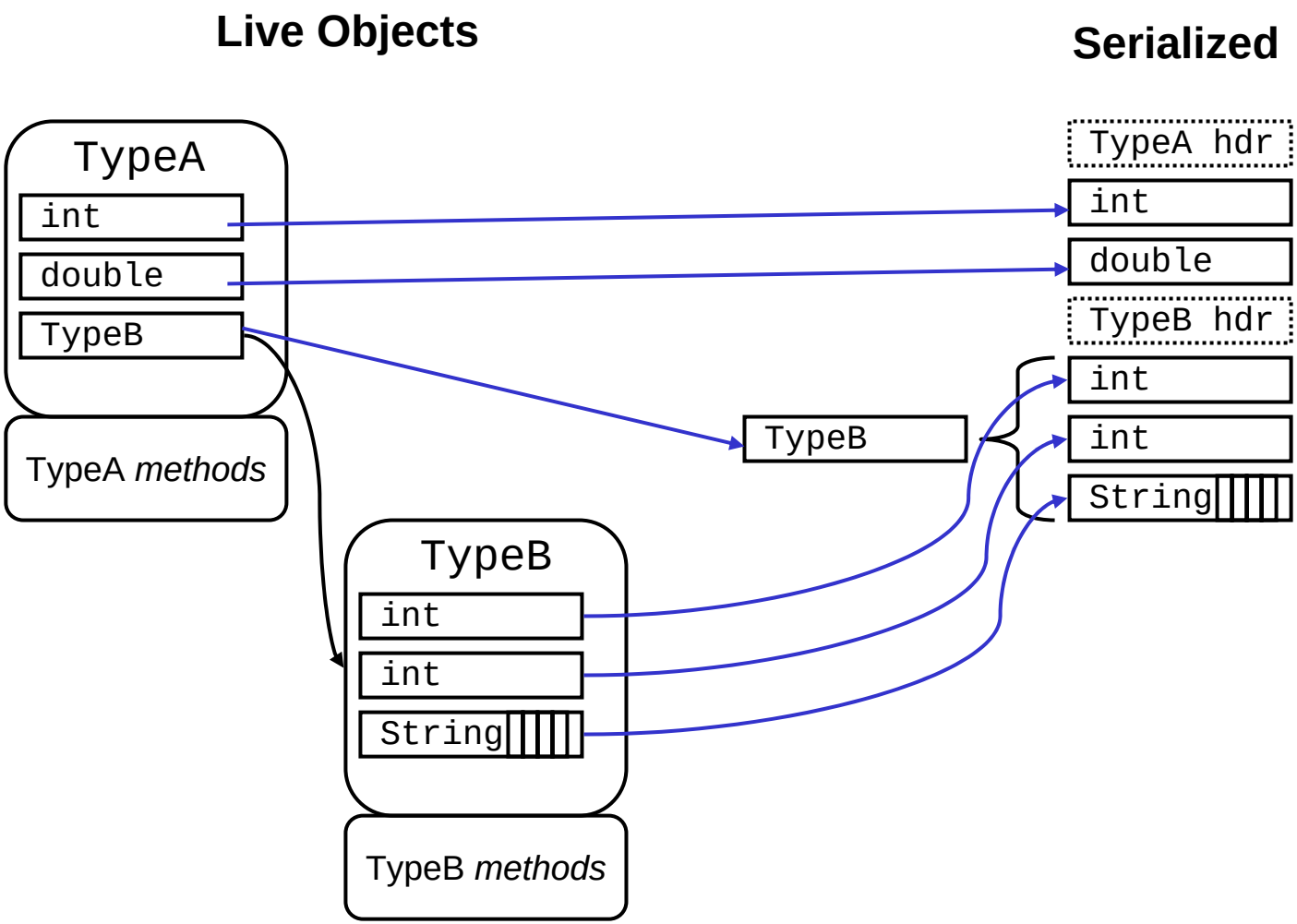
Exercise

- You are given a file tas.txt and a class TACopy.java which copies the file tas.txt to another file tas2.txt
- Modify the code so that TACopy.java copies the file tas.txt and prints each TA on a separate line
- The TAs are separated by a tab (“\t”) and are all on one line. Use StringTokenizer

Object Serialization

- Process of reading and writing objects to a stream is called object serialization
- Write objects to a stream using ObjectOutputStream and read objects to a stream using ObjectInputStream

Serialization Diagram



Serialization Example : Writing Object to an ObjectOutputstream

```
    try{
    FileOutputStream out = new
    FileOutputStream("theTime");
    ObjectOutputStream s = new
    ObjectOutputStream(out);
    SerializeExample b = new
SerializeExample();
    s.writeObject("Today");
    s.writeObject(new Date());
    s.writeObject(b);
    s.close();}
    catch(IOException e){ }
```

Serialization Example: Writing object from an ObjectInputStream

```
try{
FileInputStream in = new FileInputStream("theTime");
ObjectInputStream s = new ObjectInputStream(in);
String today = (String)s.readObject();
Date date = (Date)s.readObject();
SerializeExample ex = (SerializeExample)s.readObject();
System.out.println(ex.a);
s.close();
}
catch(IOException e)
{}
catch(ClassNotFoundException e)
{}
```

Questions

- Suppose you try to open a file for reading that does not exist. What happens?
- Suppose you try to write an object that is not serializable to the output stream. What happens?

Problem Set 10: Overview

- Read and organize the data from a specially formatted text file.
- get user input to determine which message to display next and display the next message.
- Display the response choices for each message.
- Create an output file using Java streams.
  - store the session and all messages displayed in this file.

PS10: Suggestion

- Can store the information in the Diagnostic.txt file in a tree
- Once the user specifies an answer, find the corresponding node in the tree and display the new question or answer corresponding to this choice.
- Create an output file documenting the steps the user has taken (nodes that have been visited)

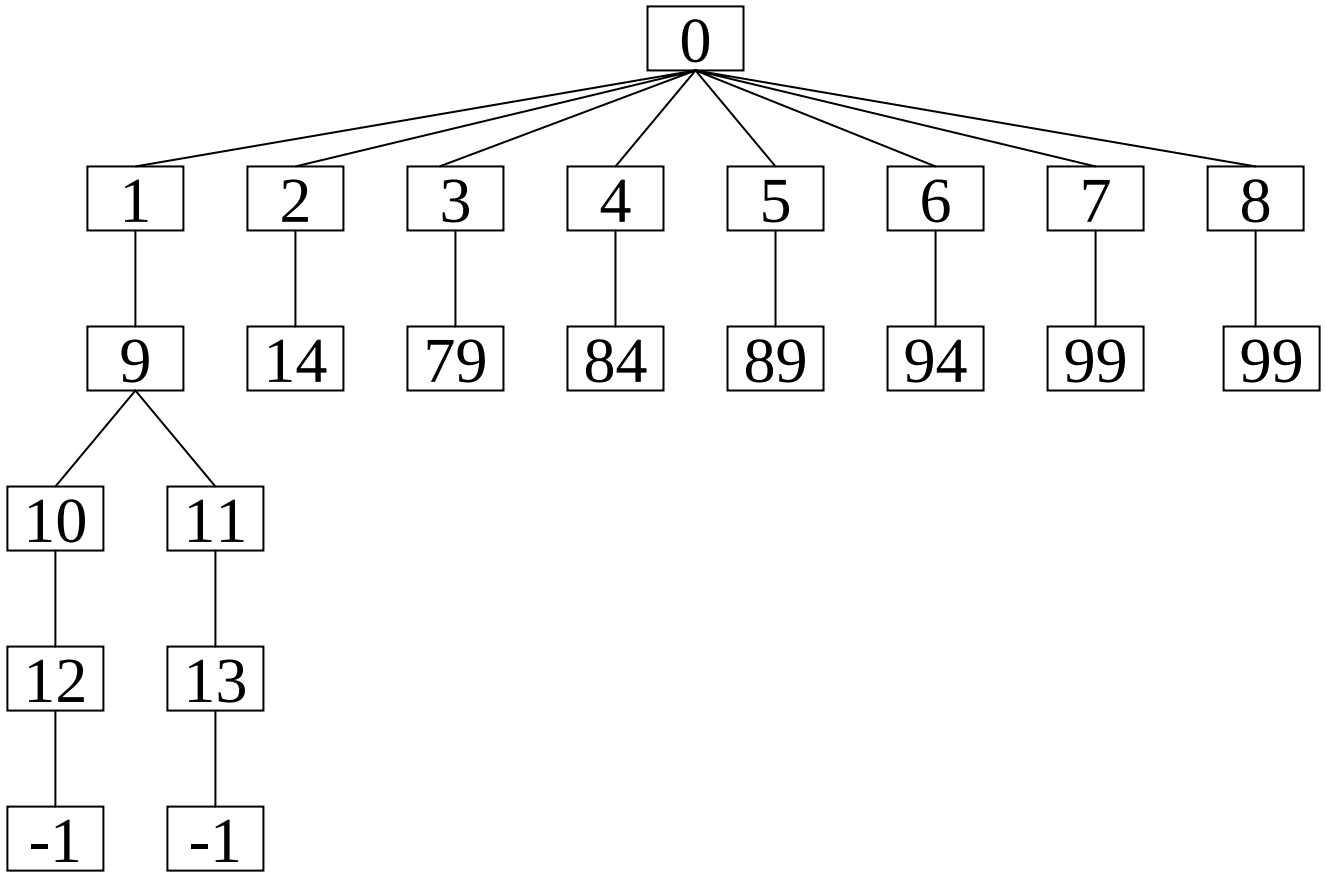
PS10: Node

- Each node has
  - ID number, message, children
  - How can you represent a node's children?
  - Methods
    - toString or some print method
    - What else would be useful?

Show text file

- Look at the text file given with the pset. Diagnostic.txt

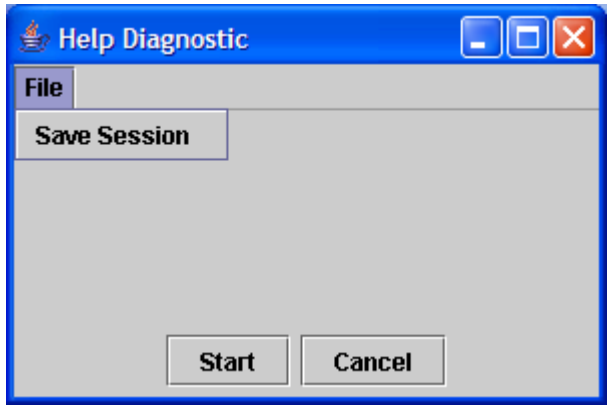
PS10: Example



PS10: Streams

- What should you look out for?
  - What exceptions should you handle?
    - FileNotFound
    - IO
    - What else?

GUI



- Look at windows XP printer troubleshooting