CS 1110
Prelim I: Review Session
Spring 2011

Exam Info

- Prelim 1: Tuesday March 8, 7:30–9:00PM
  - Uris Hall G01
- Look at the previous prelims available on course website
- Arrive early! Helps reduce stress

What's on the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluation of "new" expressions
- Evaluation/execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/Courses/cs1110/2011sp/exams/prelim1/aboutprelim1.pdf

We have a lot to "cover"

What's on the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluation of "new" expressions
- Evaluation/execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/Courses/cs1110/2011sp/exams/prelim1/aboutprelim1.pdf

Definitions

- Short answers
- Questions ask for definitions or something to be done
- Let's see 3 examples from Prelim I, spring' 07.

Definitions

- **(a) 5 pts.** What is an argument? A parameter?
- **(b) 5 pts.** What is a local variable? What is its scope?

7

Definitions

- **(a) 5 pts.** What is an argument? A parameter?
  A parameter is a variable declared in the header of a method (within the parentheses). An argument is an expression that occurs in a call of a method.
- **(b) 5 pts.** What is a local variable? What is its scope?

8

Definitions

- **(a) 5 pts.** What is an argument? A parameter?
  A parameter is a variable declared in the header of a method (within the parentheses). An argument is an expression that occurs in a call of a method.
- **(b) 5 pts.** What is a local variable? What is its scope?
  A local variable is a variable that is declared in the body of a method. Its scope begins at its declaration and continues until the end of the block in which it is declared.

9

Definitions

- **(c) 5 pts.** Explain the three steps in evaluating a new-expression —e.g. **new** Time(c,d) . The previous sentence contains an example of a new-expression, but your answer should explain what *any* new-expression does and not simply the one in the example.

10

Definitions

- **1. (c)**
- For a new-expression **new** C(…):
  - (1) Create a new object of class C;
  - (2) execute constructor call C(…);
  - (3) yield as value the name of the newly created object.
  - This question asks us to explain, later we will be asked to execute!

11

What's on the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluation of "new" expressions
- Evaluation/execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/Courses/cs1110/2011sp/exams/prelim1/aboutprelim1.pdf

12

Assignments

Assignment statement. p. 27

Execution of an assignment statement stores a value in a variable.

To execute the assignment
<var>= <expr>;
evaluate expression <expr> and store its value in variable <var>.

x= x + 1; Evaluate expression x+1 and store its value in variable x.

```
v1= 5;
v2= 3;
v1= v1 + 1
v1= v2 + v1 + 42
```

We'll execute the sequence of statements on the board.

v1 | 2

v2 | 4

13

What's on the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluation of "new" expressions
- Evaluation/execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/Courses/cs1110/2011sp/exams/prelim1/aboutprelim1.pdf

14

```
/** An instance represents a Student*/
public class Student {
  private String name; // the student's name
  private String netid; // the student's netid
  /** Constructor: a Person with  name n and netid i*/
  public Student(String n, String i) {
     name= n; netid= i;
  }
  /** set the Student's name to n */
  public void setName(String n) {
     name= n;
  }
  /** = "this Student and s have the  same netid" */
  public boolean equals(Student p) {
     return netid.equals(p.netid);
  }
```

This class is used on the next page

15

Question. Assume the following 3 assignments are executed:

```
Student p1= new Student("Bill", "bk12");
Student p2= new Student("Bill", "bk13");
Student p3= new Student("William", "bk12");
```

(a) What is the value of each of the following four expressions?

```
p1.equals(p2)        p1.equals(p3)
p1 == p2             p1 == p3
```

(b) Now consider these statements:

```
p1= new Student("Bill", "bk12");
p2= new Student("Bill", "bk13");
p3= p2;
p3.setName("Jack");
```

Evaluate "new" expressions

Below, first draw all three variables. Then execute the four statements —of course, draw any objects that are created during execution.

16

A message from Prof. Gries

17

A message from Prof. Gries

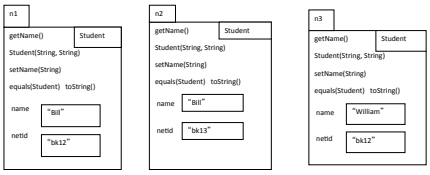
- You won't get these questions correct unless you draw the variables, objects, and execute their assignments!

18

So let's draw them…

- Student p1= **new** Student("Bill", "bk12");
- Student p2= **new** Student("Bill", "bk13");
- Student p3= **new** Student("William", "bk12");

p1 n1
p2 n2
p3 n3



19

(a) What is the value of each of the following four expressions?

p1.equals(p2)
p1.equals(p3)
p1 == p2
p1 == p3

20

(a) What is the value of each of the following four expressions?

p1.equals(p2) False
p1.equals(p3) True
p1 == p2 False
p1 == p3 False

21

(b) Now consider these statements:

p3= p2;
p3.setName("Jack");

Below, first draw all three variables. Then execute the two statements.

p1 n1
p2 n2
p3 n3

22

(b) Now consider these statements:

p3= p2;
p3.setName("Jack");

Below, first draw all three variables. Then execute the two statements.

p1 n1
p2 n2
p3 n2

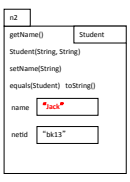
23

(b) Now consider these statements:

p3= p2;
p3.setName("Jack");

Below, first draw all three variables. Then execute the two statements.

p1 n1
p2 n2
p3 n2



24

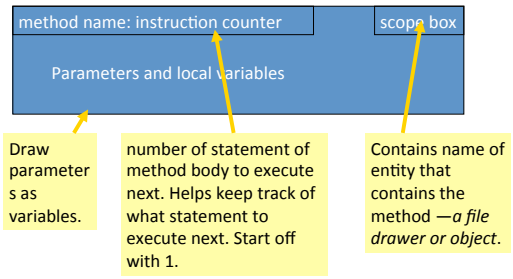
What's on the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluationof "new" expressions
- Evaluation/Execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/Courses/cs1110/2011sp/exams/prelim1/aboutprelim1.pdf

25

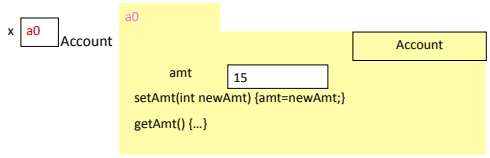
The frame (the box) for a method call

Remember: Every method is in a folder (object) or in a file-drawer.



26

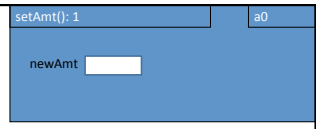
To execute the call x.setAmt(50);

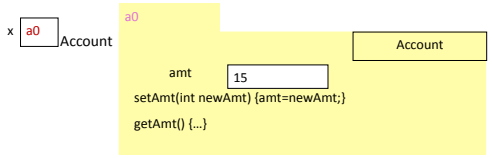


27

To execute call x.setAmt(50);

1. Draw a frame for the call.

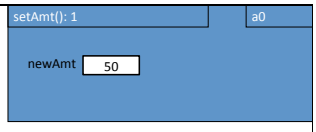


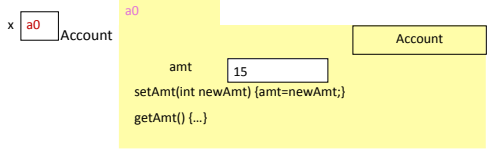


28

To execute call x.setAmt(50);

1. Draw a frame for the call.
2. Assign the value of the argument to the parameter (in the frame).

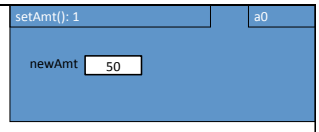




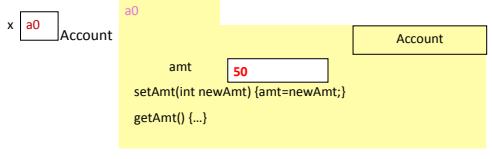
29

To execute call x.setAmt(50);

1. Draw a frame for the call.
2. Assign arg value to parameter (in the frame).



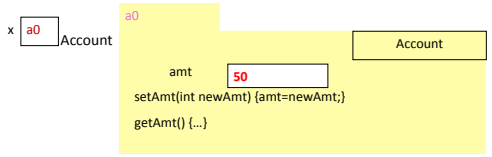
3. Execute method body. (Look for variables in the frame; if not there, look in the place given by the scope box.)



30

To execute call x.setAmt(50);

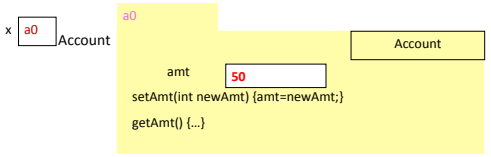
1. Draw a frame for the call.
2. Assign arg value to parameter (in the frame).
3. Execute method body. (Look for variables in the frame; if not there, look in the place given by the scope box.)
4. Erase the frame for the call.



31

To execute call x.setAmt(50);

1. Draw a frame for the call.
2. Assign arg value to par (in frame).
3. Execute method body. (Look for variables in the frame; if not there, look in the place given by the scope box.)
4. Erase the frame for the call —but not on the exam!



32

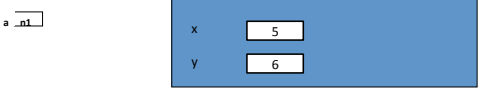
What's on the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluation of "new" expressions
- Evaluation/Execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/Courses/cs1110/2011sp/exams/prelim1/aboutprelim1.pdf

33

(d) 10 pts. Draw a frame for the call a.m(2+3, 6) of procedure m. We want to see what the frame for the call looks like after the argument values are assigned to the parameters but before the method body is executed.

```
public void m(int x, double y) {
   int z;
   z= x + y;
}
```



34

What's on the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluation of "new" expressions
- Evaluation/Execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/Courses/cs1110/2011sp/exams/prelim1/aboutprelim1.pdf

35

Return	Method	Purpose
`char`	`s.charAt(i)`	= the character at position `i` of `s`
`int`	`s.length()`	= the number of characters in `s`
`int`	`s.indexOf(n)`	= the index within `s` of the first occurrence of String `n` (`-1` if none)
`String`	`s.substring(h,k)`	= a String consisting of characters in `s[h..k-1]`, ie. `s[h]`, `s[h+1]`, ..., `s[k-1]`
`String`	`s.substring(h)`	= a String consisting of characters `s[h..s.length()-1]`

36

Q. Write function fix. This might help you: Break s into pieces, store pieces in local variables and use the pieces.

```
/** = Date s in a more suitable form.
Precondition: s contains a date in the form month/day/year,
with each part separated by "/".
Examples: 4/26/39 and 04/005/1939.
The output should be in the form year.month.day.
Examples are: 39.26.4 and 1939.04.005.
Each of day, month, and year may be any length. They
appear in exactly the same form in the input and output; just
their order and the separator are changed. */
public static String fix(String s) {
```

37

```
/**See previous slide*/
public static String fix(String s) {
    int k= s.indexOf("/");  // index of first "/"
    String month= s.substring(0,k);
    String rest= s.substring(k+1);
    k= rest.indexOf("/");  // index in rest of the only "/"
    String day= rest.substring(0,k);
    String year= rest.substring(k+1);
    return year + "." + month + "." + day;
}
```

38

What's in the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluation of "new" expressions
- Evaluation/Execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/courses/cs1110/2010fa/exams/prelim1/aboutprelim1.pdf

39

```
/** An instance represents an instrument */
public class MusicInstrument {
  private String name= null; // instrument name
  /** Constructor: an instrument with name s */
  public MusicInstrument(String s)
    {  name= s; }
  /** Constructor: an instrument with name "" */
  public MusicInstrument()
    {  name= ""; }
  /** = sound this instrument makes*/
  public String play()
    {  return "music"; }
  /** = a repr of this instrument */
  public String toString()
    { return "Instrument: " + name; }
}
```

40

```
/** An instance represents a string instrument with no name */
public class StringInstrument extends MusicInstrument {
  // number of strings on this instrument
  private int numStrings;

  /** Set the number of Strings on this  instrument to n
  public void setNumber(int n) {
        numStrings= n
  }
  /** = sound this instrument makes */
  public String play() {
        return super.toString() +  numStrings + "triiings";
  }
}
```

41

Question 2 (20 points) Write a class definition for a class PercussionInstrument that

- Is a subclass of MusicInstrument;
- Has suitable specifications on methods and definitions on fields;
- Has a field numDrums, which is the number of drums in this percussion instrument;
- Has a constructor with the name of the instrument and the number of drums as parameters;
- Overrides function play() to return the number of "druuums", similar to the way function play in class StringInstrument works.

42

1. Is a subclass of MusicInstrument;

```
/** An instance represents a percussion
  instrument */
public class PercussionInstrument
                    extends MusicInstrument{


}
```

43

2. Has a field numDrums, which is the number of drums in this percussion instrument;

```
/** An instance represents a percussion
                                instrument */
public class PercussionInstrument
                        extends MusicInstrument{

   // number of drums in this instrument
    private int numDrums;

}
```

44

3. Has a constructor with the name of the instrument and the number of drums as parameters;

```
/** An instance represents …*/
public class PercussionInstrument
                    extends MusicInstrument{

   // number of drums in this instrument
    private int numDrums;

}
```

45

```
/** An instance represents an
    instrument */
public class MusicInstrument {
  // Member of Congress' name
  private String name= null;
  /** Constructor: an instrument with
      name s */
  public MusicInstrument(String s) {
        name= s;
  }
  /** Constructor: an instrument with
      name "" */
  public MusicInstrument() {
        name= "";
  }
   /** = sound this instrument makes*/
  public String play() {
        return "music ";
  }
    /** = a repr of this instrument */
  public String toString() {
        return "Instrument: " + name;
  }
}
```

```
/** An instance represents a string
instrument with no name */
public class StringInstrument extends
      MusicInstrument {
  /** number of strings on this
      instrument */
  private int numStrings;

  /** Set the number of Strings on this
instrument to n
  public void setNumber(int n) {
        numStrings=  n
  }
  /** = sound this instrument makes */
  public String play() {
        return super.toString() +
        numStrings + "triiings";
  }
}
```

46

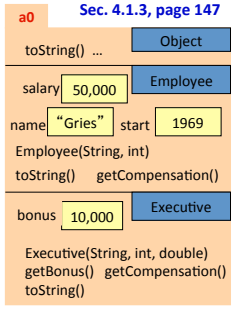
```
public class Executive extends Employee {
  private double bonus;
  /** Constructor: name n, year hired
          d, salary 50,000, bonus b */
  public Executive(String n, int d, double b) {
     super(n, d);
     bonus= b;
   }
}
```

The first (and only the first) statement in a constructor has to be a call to a constructor of the superclass.

Principle: Fill in superclass fields first.

Calling a superclass constructor from the subclass constructor

Sec. 4.1.3, page 147



47

3. Has a constructor with the name of the instrument and the number of drums as parameters;

```
/** An instance represents a percussion instr */
public class PercussionInstrument
                        extends MusicInstrument{
   // number of drums in this instrument
   private int numDrums;
   /** Constructor: an instance name s with n drums */
   public PercussionInstrument(String s, int n) {
         super(s);
         numDrums= n;
    }
}
```

48

4. Overrides function play() to return the number of "druuums", similar to the way function play in class StringInstrument works.

```
/** An instance represents a percussion instrument */
public class PercussionInstrument extends MusicInstrument{

    // number of drums in this instrument
    private int numDrums;

    /** Constructor: an instance name s with n drums */
    public PercussionInstrument(String s, int n) {
        super(s);
        numDrums= n;
    }

}
```

49

```
/** An instance represents an
    instrument */
public class MusicInstrument {
  // Member of Congress' name
  private String name= null;

  /** Constructor: an instrument with
      name s */
  public MusicInstrument(String s) {
        name= s;
  }
  /** Constructor: an instrument with
      name "" */
  public MusicInstrument() {
        name= "";
  }
   /** = sound this instrument makes*/
  public String play() {
        return "music ";
  }
    /** = a repr of this instrument */
  public String toString() {
       return "Instrument: " + name;
  }
}
```

```
/** An instance represents a string
instrument with no name */
public class StringInstrument extends
      MusicInstrument {

  /** number of strings on this
      instrument */
  private int numStrings;

  /** Set the number of Strings on this
instrument to n
  public void setNumber(int n) {
          numStrings=  n
  }
  /** = sound this instrument makes */
  public String play() {
          return super.toString() +
          numStrings + "triiings";
  }
}
```

50

Purpose of super and this **Sec. 4.1, pages 144-145**

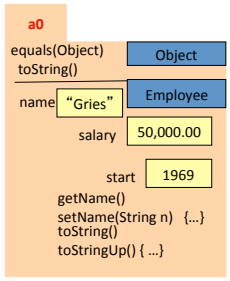
this refers to the name of the object in which it appears.
super is similar but refers only to components in the partitions above.

```
/** = String representation of this
Employee */
public String toString() {
    return this.getName() + ", year " +
        getStart() + ", salary " + salary;
}
```

ok, but unnecessary

```
/** = toString value from superclass */
public String toStringUp() {
    return super.toString();
}
```

necessary



51

```
/** An instance represents an
    instrument */
public class MusicInstrument {
  // Member of Congress' name
  private String name= null;

  /** Constructor: an instrument with
      name s */
  public MusicInstrument(String s) {
        name= s;
  }
  /** Constructor: an instrument with
      name "" */
  public MusicInstrument() {
        name= "";
  }
   /** = sound this instrument makes*/
  public String play() {
        return "music ";
  }
    /** = a repr of this instrument */
  public String toString() {
       return "Instrument: " + name;
  }
}
```

```
/** An instance represents a string
instrument with no name */
public class StringInstrument extends
      MusicInstrument {

  /** number of strings on this
      instrument */
  private int numStrings;

  /** Set the number of Strings on this
instrument to n
  public void setNumber(int n) {
          numStrings=  n
  }
  /** = sound this instrument makes */
  public String play() {
          return super.toString() +
          numStrings + "triiings";
  }
}
```

52

5. Overrides function play() to return the number of "druuums", similar to the way function play in class StringInstrument works.

```
/** An instance represents a percussion instrument */
public class PercussionInstrument extends MusicInstrument{

    // number of drums in this instrument
    private int numDrums;

    /** Constructor: an instance name s with n drums */
    public PercussionInstrument(String s, int n) {
        super(s);
        numDrums= n;
    }

    /** = sound this instrument makes */
    public String play() {
        return super.toString() + numDrums + "druuums";
    }
}
```

53

6. Has suitable specifications on methods and definitions on fields

```
/** An instance represents a percussion instrument */
public class PercussionInstrument extends MusicInstrument{

    // number of drums in this instrument
    private int numDrums;

    /** Constructor: an instance name s with n drums */
    public PercussionInstrument(String s, int n) {
        super(s);
        numDrums= n;
    }

    /** = sound this instrument makes */
    public String play() {
        return super.toString() + numDrums + "druuums";
    }
}
```

54

6. Has suitable specifications on methods and definitions on fields Done!

```
/** An instance represents a percussion instrument */
public class PercussionInstrument extends MusicInstrument{

    // number of drums in this instrument
    private int numDrums;

    /** Constructor: an instance name s with n drums */
    public PercussionInstrument(String s, int n) {
        super(s);
        numDrums= n;
    }

    /** = sound this instrument makes */
    public String play() {
        return super.toString() + numDrums + "druuums";
    }
}
```

55

What's n the exam?

- Definitions of terms and key concepts
- Execution of assignment statements
- Evaluation of "new" expressions
- Evaluation/Execution of method calls
- Execute sequence of statements
- String functions
- Writing class definitions
- See "About the prelim" on course website
  http://www.cs.cornell.edu/Courses/cs1110/2011sp/exams/prelim1/aboutprelim1.pdf

56

Question period

57

 Good Luck! 

58