

CSE 113 A

January 25 – 29, 2010

Announcements (1/25)

- ⚙ If you have not picked up a syllabus, please do so
- ⚙ Assignment #1 – sign and return form on last page of syllabus – must be turned in by end of class Monday, January 25th to receive full credit.
- ⚙ Engineering Accounts Reminder



Announcements (1/27 & 1/29)

- ⊗ Engineering Account Problems
- ⊗ Source code will be posted on the schedule page as well. To use it yourself, you will need to download the file and then unzip it. Then, you can ask Greenfoot to open that scenario.
- ⊗ Web-CAT accounts
- ⊗ Exam 1 is Wednesday 2/3 – Review Sheet posted – will be reviewed 2/1 in class



The screenshot shows a Java IDE window titled "Class Edit Tools Options" with a menu bar (Compile, Undo, Cut, Copy, Paste, Find, Find Next, Close) and a "Source Code" button. The code is as follows:

```

import greenfoot.*; // World, Actor, GreenfootImage, Greenfoot and MouseClick

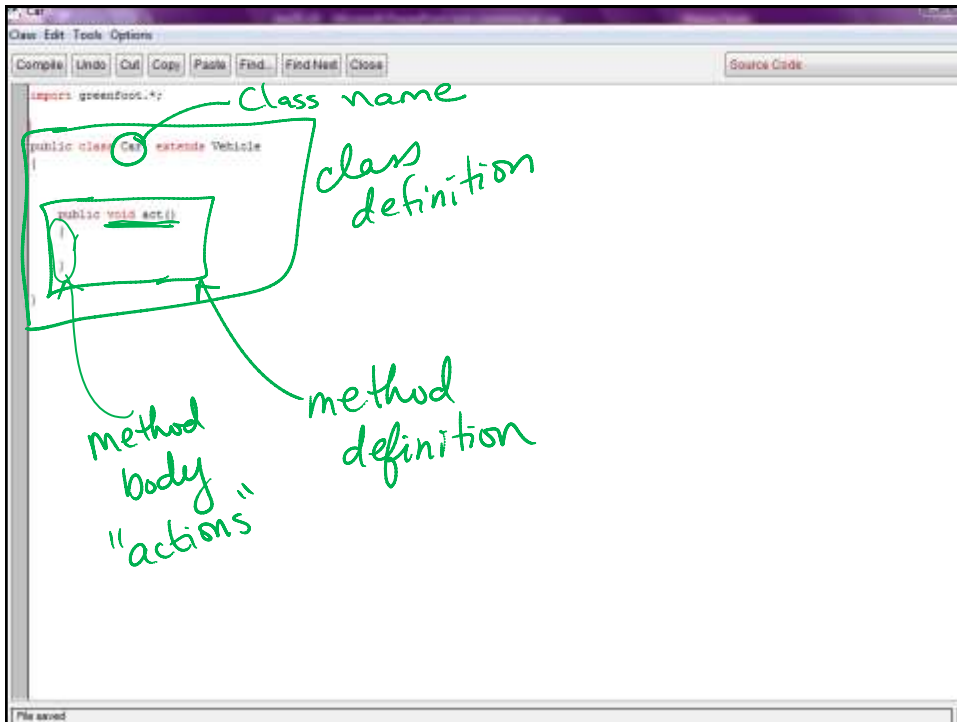
/**
 * Write a description of class Car here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Car extends Vehicle

/**
 * Act - do whatever the Car wants to do. This method is called whenever
 * the 'Act' or 'Run' button gets pressed in the environment.
 */
public void act()
{
}

```

Handwritten annotations in green ink:

- A red oval highlights the import statement: `import greenfoot.*;`
- A green box highlights the header comment block: `/** ... */`
- A green box highlights the `act()` method comment: `/** ... */`
- Arrows point from the text "Comments" to the header comment and the `act()` comment.
- Arrows point from the text "Programmer's notes - to be read by humans" to the header comment and the `act()` comment.
- An arrow points from the text "Header comment Demographic information" to the header comment.



A screenshot of a code editor window showing a Java class definition. The code is as follows:

```
import greenfoot.*;  
  
public class Car extends Vehicle  
{  
    public void act()  
    {  
        // ...  
    }  
}
```

Handwritten annotations in green ink identify parts of the code:

- "Class name" points to `Car` in `public class Car`.
- "class definition" points to the entire `public class Car` line.
- "method definition" points to the `public void act()` line.
- "method body 'actions'" points to the curly braces and content of the `act()` method.

Selection using if-statements

Syntax:

```
if ( condition )  
{
```

actions to happen when condition
is true

```
}
```



```
if (atWorldEdge ( ) )  
{  
    turn (14);  
}
```



7

```
if (random number is in the  
    range we want)  
{  
    explode!  
}
```



8


Ex)

Greenfoot. getRandomNumber(5);

tells Java
where to find
the method


calling a method

limit/
range



9

If limit on getRandomNumber is set
at 5, we get numbers back in the
range 0-4.



10

Eg) "20% of the time"
 Greenfoot.getRandomNumber(100) \leq 20

- returns true if number $<$ 20
- returns false otherwise

11



Operators

Symbols that ~~do~~ tell Java to perform certain actions. Most often, operators produce a result.

$<$ "less than" \leq "less than or equal to"

$>$ "greater than" \geq "greater than or equal to"

12



$==$ "equals"
 $!=$ "not equal"

$+$ "addition"
 $-$ "subtraction"
 $*$ "multiplication"
 $/$ "division"

13



Random numbers:

$x = \text{limit} - 1$

$0 - x$

We want numbers:


$-y$ to y

14



real range is
 $0 - \cancel{29}^{30}$

want:
 range:
 -15 to 15

15 

real range
 $0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \dots$

wanted
 range
 $-15 \ -14 \ -13 \ -12 \ -11 \ -10 \ -9 \ -8 \ -7 \ -6$

16 