

WES-CS GROUP MEETING #6

Exercise 1: Boolean conditions

The Child class defined on the next page was inspired by the following three short poems:

What is the matter with Liza Jane?
She's perfectly well and she hasn't a pain,
and it's lovely rice pudding for dinner again.
What *is* the matter with Liza Jane?

Christopher Robin had sneezles and wheezles,
they bundled him into his bed.
They gave him what goes
for a cold in the nose
and much more for a cold in the head.

There was a little girl
who had a little curl
right in the middle of her forehead.
And when she was good
she was very, very good;
but when she was bad
she was horrid!

```

class Child {
    /** data members */
    private String name;
    private boolean hasPain;
    private boolean hasSneezles;
    private boolean hasCurl;

    /** constructor */
    public Child(String aName) {
        name = aName;
        hasPain = hasSneezles = hasCurl = false;
    }

    /** mutator methods */
    public void setSneezles() {
        hasSneezles = true;
    }

    public void setCurl() {
        hasCurl = true;
    }

    /** other public methods */
    public boolean needsDoctor() {
        return ( hasSneezles || hasPain );
    }

    public boolean actsBadly( boolean isGood, boolean ricePudding ) {
        return ((hasCurl && !isGood) || ricePudding);
    }
}

```

Part (a): Draw memory diagrams to show the effect of executing each of the following lines of code.

```
Child liza, chris, girl;  
  
liza = new Child("Liza Jane");  
  
chris = new Child("Christopher Robin");  
  
girl = new Child("girl");  
  
chris.setSneezles();  
  
girl.setCurl();
```

Part (b): Figure out what value is returned by each of the following method calls. A good way to do this is to write down the returned expression (e.g., a call to *needsDoctor* returns (*hasSneezles* // *hasPain*)), then replace the identifiers with their (boolean) values, then evaluate the resulting boolean expression.

```
liza.needsDoctor();  
  
chris.needsDoctor();  
  
girl.needsDoctor();  
  
liza.actsBadly(true, true);  
  
chris.actsBadly(true, false);  
  
girl.actsBadly(true, false);
```

Exercise 2: Boolean conditions

Play a game using the decks of yellow and blue cards (for the cards, print pages 8–10 on blue paper, print pages 11–13 on yellow paper, and cut out the individual cards). Each blue card has an English sentence that is either true or false. Each yellow card has one of three symbols: !, &&, or ||. The game is played as follows:

- Each player starts with 6 cards, 3 yellow and 3 blue.
- When it's your turn, if you can make a logical formula that evaluates to true using at least 4 of your cards, put down the formula and draw new cards. Otherwise, trade in one of your cards for a card of the same color.
- The first person to put down at least 12 cards wins.

Don't forget the precedences of the logical operators:

!	highest precedence
&&	next highest precedence
	lowest precedence

Exercise 3: The Code game

An incomplete CodeGame class is on the next two pages.

First, get some practice with code-tracing:

- Break up into two teams.
- Each team traces the code (starting with the *main* method), using a real die and a real coin for the *roll* and *toss* methods. The team with the most points wins.

Now work together to write the actual code for the *roll* and *toss* methods, then run the program twice (once for each team). Who won this time??

Remember that you can generate pseudo-random numbers using the *random* method of the *Math* class:

```
public static double random()
```

Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

```

class CodeGame {
    // data members
    private int points; // the number of points
    private static final boolean HEADS = true; // for coin toss
    private static final boolean TAILS = false; // for coin toss

    // constructor
    public CodeGame() {
        points = 0;
    }

    // getPoints: return the current number of points
    public int getPoints() {
        return points;
    }

    // roll: return a number 1 - 6 (representing a roll of a die)
    private int roll() {
        // not yet implemented
        return 1;
    }

    // toss: return HEADS or TAILS representing a coin toss
    private boolean toss() {
        // not yet implemented
        return HEADS;
    }

    // main method
    public static void main( String[] args ) {
        CodeGame game = new CodeGame();
        game.play();
        System.out.println( "Your score is " + game.getPoints());
    }
}

```

```

// play: play the game
public void play() {
    int oneRoll; // the current roll of the die; 1 - 6
    boolean oneToss; // the current toss of the coin; HEADS or TAILS

    oneRoll = roll();
    while (oneRoll <= 3) {
        points++;
        oneRoll = roll();
    }

    if ( oneRoll == 6 || oneRoll == 4) points++;
    else points--;

    oneRoll = roll();
    for (int i = oneRoll; i > 0; i--) {
        points = points + 2;
    }

    oneToss = toss();
    if ( oneToss==HEADS ) points = points*2;
    else points = points + 2;

    do {
        oneRoll = roll();
        points = points - oneRoll;
        oneToss = toss();
    } while ( !(oneToss == HEADS) );

    oneRoll = roll();
    oneToss = toss();
    if ( (oneToss == HEADS) && oneRoll <= 3 ) points = points + 12;
    else if ( (oneToss == HEADS) && oneRoll > 3 ) points = points + 9;
    else if ( !(oneToss == HEADS) && oneRoll <= 3) points = points + 6;
    else points = points + 3;

    oneRoll = roll();
    switch ( oneRoll ) {
    case 1:  points = points + points / 2;
            break;
    case 2:  points = points + roll();
            break;
    case 3:  oneToss = toss();
            if ( oneToss == HEADS ) points = points + 10;
            break;
    case 4:  points = points + 4;
            break;
    case 5:  points = points + roll() * oneRoll - oneRoll * 2;
            break;
    case 6:  points = points - oneRoll - roll();
            }
    }
}

```

The Earth is flat

Madison is the capital of Wisconsin

In Java, * has higher precedence than +

In Java, + has higher precedence than -

The moon is made of green cheese

17 is a prime number

There is not a DDR machine in Memorial Union

The dorms get Cartoon Network

Red and blue make purple

A chestnut tree makes acorns

A giraffe has a black tongue

Ducks can't fly

We are learning C++ in CS302

A soccer ball is round

The sun moves around the Earth

The UW-Madison was founded in 1492

There are 7 WES-CS groups

This room is CS 3817

Miss Muffet is afraid of spiders

The time now is 3:33pm

The sun sets in the west

Java programs start executing in a class named Main

The following code will compile: `boolean b = "true";`

An if statement will repeat until the condition is false

&&

&&

&&

&&

&&

&&

&&

&&

&&

&&

&&

&&

||

||

||

||

||

||

||

||

||

||

||

||

!

!

!

!

!

!

!

!

!

!

!

!