# Introduction to Programming

Last Year's Practice Project



## Last year's practice project

- The next few slides reproduce the problem description and suggested procedure from the handout
- The project description is taken from the 1999 text by Duane A Bailey and Duane W Bailey, *Java Elements: Principles of Programming in Java.* (McGraw-Hill Education)



## The Problem

- Baby, Crab and Cone Puzzle
  - You are to develop a program to validate potential solutions to a classic puzzle recast as Mama's puzzle of the baby, crab, and cone. As the story goes, Mama headed off to the ice cream shop on a hot summer day with her delightful baby and pet crab. On the way she crossed a river with the aid of a ferry boat that, rowed by herself, could carry one other. A little thought demonstrates that this was no minor obstacle—it required three trips in the ferry to get everyone across (the crab it seems, was no great swimmer). By the time she got to the shop, she was clearly ready to make a purchase.
    - The farmer's puzzle of the chicken, fox and the corn



## The Problem continued

- Having purchased the largest cone possible, Mama set back toward home with all in tow: baby, crab, and cone. At the river, however, she faced a dilemma: in what order should she row everyone across in the ferry? The difficulty was that since only one of the baby, crab, or cone could be ferried across (it must have been a huge cone), two must be left behind. Unfortunately, if the baby was left unattended with the crab, violence of some form would surely ensue. If the baby was left with the cone, it would overindulge. Thus, each trip of the ferry had to be made with care, making sure that the baby was never left with either of the other two on same shore.

## The Problem continued

- To facilitate Mama's journey, you are to write a planning program of sorts, that allows the harmless simulation of the various possibilities. We have in mind a program with output similar to the following:

```
Welcome to the Puzzle of the Baby, the Crab, and the Cone.
One hot summer day, Mama crossed the river to the left bank
take a break from programming and buy an ice cream. She
brought her baby and her pet crab. On the return she came to
the river and was faced with a dilemma: how could they all
cross without disaster? The ferry holds Mama and just one
other item. The problem is that
* if the baby is left with the cone, it spoils its dinner.
* if the baby is left with the crab, the crab bites the baby.
```



## The Problem continued

```
Please help Mama make her journey across
the river!

Mama is on the left.
The baby is on the left bank.
The crab is on the left bank.
The cone is on the left bank.
Who would you like Mama to cross with?
(type 0 for no one, 1 for baby, 2 for
crab, or 3 for cone)
```



## The Problem continued

```
(type 0 for no one, 1 for baby, 2 for crab, or 3
for cone)
```
**2**
```
Mama crosses the river with her pet crab.
Mama is on the right.
The baby is on the left bank.
The crab is on the right bank.
The cone is on the left bank.
Oh no! The baby ate the ice cream.
We'll have to stop and help Mama and her family.
Undoubtedly, she'll need your help again.
```



## The Problem continued

- As you can tell, the program keeps track of the location of each of the parties, perhaps by a boolean. After describing the state of the situation, the facilitator is asked to determine who rides in the ferry with Mama. As Mama leaves shore, the various lethal conditions are checked for and the simulation is potentially terminated. If, however, everyone eventually makes it to the destination (here, the "right bank"), the puzzle is solved. A good program would point out that the solution might have been faster if more than seven moves were required.

# Procedure

This is quite a complex program, so the following steps should be considered in the design of your program. Once designed, the program is more easily coded:

- Identify all the variables that will be needed. Variables help to keep track of the state of the program. If something must be remembered, it must be accounted for as a variable. For example, it might be useful to keep track of the number of times the ferry has crossed the river.
- For each variable, identify its type, and if necessary, its initial value. This should complete your declarations.



# Procedure continued

- Break the program down into several sizeable pieces. It's not important to know immediately how each piece is implemented, but it is necessary to identify logically distinct portions of the program. For example, the instructions must be written at the beginning, the results at the end, and in the middle it is necessary to print the location of all the participants. There may be other parts, as well.
- Identify those parts of the program that are part of a conditional or looping construct, remembering the features that distinguish each of the loop types.
- Our informal design is completed if a reasonable person can follow the logical states of the program without doing anything that seems contrary to the imagined execution of the program.



# Procedure continued

- Armed with a design, it is now possible to implement each of the logical components in a programming language.
  - It is often the case that the design has to be augmented to take into account the finer details. As you implement each of the logical components of the program, you may find it useful to test to see if the program works as expected. For example, you run the program to see if the instructions and initial state is printed correctly. The intermediate steps also help to ward off any unforeseen problems as early as possible.



# Procedure continued

- Once completed, test your program thoroughly.
  - Does it work in short simulations (as seen in the one above)?
  - Does it work when you follow the optimal solution?
  - Does it work, even if Mama makes far too many trips?
  - What happens if the number 4 is provided as input to the question?

# Steps...

- Identify all the variables that will be needed.

We will need a variable for each of Mama, Baby, Crab and Cone to keep track of which bank they are on.

The handout suggests possibly using **boolean** variables – we could use **false** to mean the character is on the left bank and **true** to mean it is on the right bank. To simplify the output messages, however, we will use two String constant values, “LEFT” and “RIGHT” to keep track of the characters’ locations.

To declare a constant, use the word, **final**, in front a variable declaration to indicate that its value cannot be changed after it is initialised.



# Variables needed...

```
// constants
final String LEFT = "Left";
final String RIGHT = "Right";

// variables to keep track of the location of
// Mama and the other characters
String bankMama = LEFT;
String bankBaby = LEFT;
String bankCone = LEFT;
String bankCrab = LEFT;
```



# Variables needed...

- Identify all the variables that will be needed.

We will also need a variable to store the user’s option in deciding who crosses the river with Mama.

```
int option = 0; // option that the user chooses
```

As hinted in the handout, we can also declare a variable to count how many times the ferry crosses the river.

```
int crossings = 0; // count ferry crossings
```



# Steps...

- Break the program down into several sizeable pieces.

Display the instructions

Get the user input and update where the characters are until the puzzle is solved or disaster strikes

Display the outcome

# Steps...

- Identify those parts of the program that are part of a conditional or looping construct, remembering the features that distinguish each of the loop types.

Get the user input and update where the characters are until the puzzle is solved or disaster strikes

- Mama is going to cross the river at least once so we will use a do...while loop for the main body of the program



# The program loop...

```
do {
   Display where the characters are
   Display options to user
   Get user input
   IF input is invalid THEN
      Display error message
   ELSE
      Update character locations
   Check if disaster has struck
} while (no disaster and puzzle not solved);
```



# Getting the user input...

- We will display a prompt to the user indicating the options in terms of who crosses the river with Mama
- To simplify the interface the same options will be offered each time round the loop
- We will define a set of constants to represent each option

```
final int MAMA = 0; // to select mama only
final int BABY = 1; // to select crossing with baby
final int CRAB = 2; // to select crossing with crab
final int CONE = 3; // to select crossing with cone
```

- We will use these constant values in a switch statement when processing the user input



# Further refinements

```
IF input is invalid THEN
```

- The user input is invalid if they have not chosen one of the four values, 0 to 3, or if the character to take across the river with Mama is not on the same bank as she is

```
Check if disaster has struck
```

- Disaster has struck if the baby is on the same bank as the crab or the cone and Mama is not also on that bank

# The program structure

- We could provide a separate method to display the instructions and another one to display the final outcome
- For now, we will just put all the code in a single static method named, solvePuzzle()



# The program...

```
public static void solvePuzzle() {

   // constants
   final String LEFT  = "Left";
   final String RIGHT = "Right";

   // constants to select mama crossing
   final int MAMA = 0; // on her own
   final int BABY = 1; // with baby
   final int CRAB = 2; // with crab
   final int CONE = 3; // with cone
```



# The program...

```
// Variables to keep track of the location of
// Mama and the other characters.
// Initially, they are all on the left bank.
String bankMama = LEFT;
String bankBaby = LEFT;
String bankCone = LEFT;
String bankCrab = LEFT;

// Variables relating to user input
int option; // option that the user chooses
boolean valid; // option chosen is valid

boolean disaster = false; // has disaster struck?
```



# The program...

```
// Display opening message
TextIO.put("Welcome to the Puzzle of the Baby, the");
TextIO.putln("Crab, and the Cone.");
TextIO.put("One hot summer day, Mama crossed the river");
TextIO.putln(" to the left bank to take a break from");
TextIO.put("programming and buy an ice cream. She ");
TextIO.putln("brought her baby and her pet crab.");
TextIO.put("On the return she came to the river");
TextIO.putln("and was faced with a dilemma:");
TextIO.put("how could they all cross without ");
TextIO.putln("disaster? The ferry holds Mama");
TextIO.put("and just one other item.");
```

# The program...

```
TextIO.putln(" The problem is that:");
TextIO.put(" * if the baby is left with the cone, ");
TextIO.putln("it spoils its dinner.");
TextIO.put(" * if the baby is left with the crab, ");
TextIO.putln("the crab bites the baby.");
TextIO.put("Please help Mama make her journey ");
TextIO.putln("across the river!");
```



# The program loop...

```
do {
   // Display location of mama and items and offer menu
   TextIO.putln();
   TextIO.putln("Mama is on the " + bankMama);
   TextIO.putln("The baby is on the " + bankBaby + " bank");
   TextIO.putln("The crab is on the " + bankCrab + " bank");
   TextIO.putln("The cone is on the " + bankCone + " bank");

   TextIO.putln("Who would you like Mama to cross with?");
   TextIO.put("(type 0 for no one, 1 for baby, ");
   TextIO.putln("2 for crab, 3 for cone");
```



# The program loop continued

```
// Get the user choice
option = TextIO.getlnInt();

// Check that choice is valid
switch(option) {
    case MAMA: valid = true; break;
    case BABY: valid = bankBaby == bankMama; break;
    case CRAB: valid = bankCrab == bankMama; break;
    case CONE: valid = bankCone == bankMama; break;
    default: valid = false;
}
```



# The program loop continued

```
if (!valid) {
  if (option >= MAMA && option <= CONE) {
     TextIO.put("That is on the other bank, ");
     TextIO.putln("Mama can't take it!");
  } else {   // user input error
     TextIO.putln("That wasn't an option!");
  }
} else {  // choice is valid so process it
      if (bankMama == LEFT) {
         bankMama = RIGHT;
     } else { // Mama was on right bank
         bankMama = LEFT;
     }
```

# The program loop continued

```
    crossings++;
    TextIO.put("Mama has crossed the river");

    switch(option) { // update character locations
        case MAMA: TextIO.putln(); break;
        case BABY: TextIO.putln(" with the baby");
                   bankBaby = bankMama; break;
        case CRAB: TextIO.putln(" with the crab");
                   bankCrab = bankMama; break;
        case CONE: TextIO.putln(" with the cone");
                   bankCone = bankMama; break;
    } // end switch
```



# The program loop continued

```
        disaster = (bankBaby != bankMama) &&
                   ((bankBaby == bankCrab) ||
                    (bankBaby == bankCone));
      } // end if

    // Continue until disaster or until all four characters are
    // on the right bank
    } while (!disaster &&
            !((bankBaby == RIGHT) && (bankCrab == RIGHT) &&
                (bankCone == RIGHT) && (bankMama == RIGHT)));
```



# Display the outcome

```
if (!disaster) {
   TextIO.putln();
   TextIO.put("OH THANK YOU!!! Mama, baby, ");
   TextIO.putln("crab and cone are all safely across!");
   if (crossings > 7) {
      TextIO.put("But poor Mama had to cross the river ");
      TextIO.putln(crossings + "times:");
      TextIO.putln("she's ready for that ice cream!!");
   if (crossings > 11) {
      TextIO.put("OH NO!! ");
      TextIO.putln("The ice cream has melted!!!");
   }
}
```



# Display the outcome

```
  } else {  // disaster has struck
    if (bankBaby == bankCone) {
        TextIO.putln("Oh no!! The baby ate the ice cream!");
    }
    if (bankBaby == bankCrab) {
        TextIO.put("Oh no! The baby is crying!!");
        TextIO.putln("The crab has bitten the baby!!");
    }
    TextIO.put("We'll have to stop and help Mama");
    TextIO.putln(" and her family");
    TextIO.putln("Undoubtedly, she'll need your help again."
  } // end if

} // end solvePuzzle()
```

# The program is on Moodle

- Download it in the lab and run it a few times to test it
  - Include cases where disaster strikes
  - Include cases where mama, baby, crab and cone all make it safely across the river
- There will be a practice programming project to do over the Christmas break