Christophe Cornu

Fun Science
With Your Computer

Original Edition
Last modified: 2007/7/14
www.funsciencewithyourcomputer.org

Table of Content

Welcome note

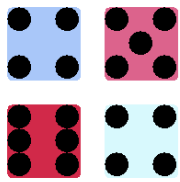
Science and technology are fun to learn when you play with them. It's all about imagination, discovery, trial and doing things with others.

How does a computer play chess? How can you create a space game that uses the laws of gravity? Create a maze and find a way out? How about a self-learning language translator? Throw a dice to calculate the digits of Pi 3.141592653… Yes, that's right. You will achieve all of the above. You will run, play with and improve computer programs designed specifically to help you understand the big picture.

This book is for teenagers (grade six and above), students, parents and teachers. If you have no programming experience you will need a friend or a mentor familiar with Java programming to get your computer setup and ready to run the programs. How to run the programs in this book will help you getting started.

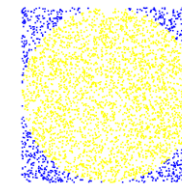
Each activity is like a cooking recipe producing a surprising and unique meal. Our eyes open up to a key scientific puzzle that once teased the best minds throughout the centuries. First we follow the recipe – program the computer. Then we enjoy the meal and share it with friends – we run the program and show it to our friends. And then we refine it – we take on the guided exercises proposed in the activity. We add our own improvements to the recipe - the program. Each activity highlights related major dates and famous people. We get a sense of the amazing evolution of ideas and inventions throughout our history – with more yet to come.

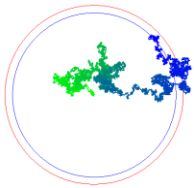
Activities can be explored in the order of your interest.



Simulate a dice with Activity 1 – Pseudo Random. Did you know it is very hard to have truly random numbers? Once we know how to generate numbers that apparently look random, we will learn how to do very interesting things with them.

What can you do with one random number? Or with two random numbers? Not much. Be prepared to be surprised. You can do nearly everything with a million of them. Activity 2 - Monte Carlo illustrates how the digits of Pi can be calculated by randomly picking points inside a square.



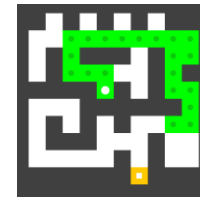


Once you manipulate millions of random numbers, you can predict average behaviors. For example, if you walk randomly changing direction after each step, you can estimate the distance you are from your starting point. Can you guess what that distance is? Activity 3 - Random Walk introduces you to something very dear to biologists and physicists.

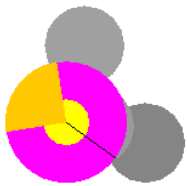
A newborn baby does not know much about life on Earth. What a human baby is incredibly good at is learning – learning how to see, how to move, how to communicate. Activity 4 - Translator is a program that initially knows nothing except how to ask users to complete information it is missing. You can teach it any language you can think of.

In 2006 Joshua Plotnik showed elephants pass the mirror test. Elephants, along with humans, dolphins and great apes, can recognize themselves in a mirror. Well, Activity 5 - Launcher is a program that knows how to look upon itself to discover what it can do. You can use it to start any of the programs from this book. And your own.

How does a postman decide the order in which mail is to be delivered? The postman doesn't. A computer searches the most efficient trip for the postman. Just the same way you plan your vacation road trip and find the best path from your home to your five star hotel. Activity 6 – Path Finder can find any path in and out of any map.



Consider a recipe that requires three hours to make a cake for four people. Can you improve the recipe so the cake is just as good but requires less than sixty minutes of preparation? Activity 7 - Slow or Fast investigates two ways to resolve a mathematical problem. One is constantly fast. The other is increasingly slow. Be a good cook – er, a good programmer.



We all worry about weight or have relatives who worry about theirs. Well it is time to have a little recreation with mass, gravity and Mr. Isaac Newton. Activity 8 – Space Simulation is a multiplayer space simulation action game realistically based on the laws of Physics.

We spend more time on the Internet than we do watching

TV. Have you ever wondered how a web browser works? Activity 9 - Web Browser shows how we can create a very simple text based web browser by loading content and navigating through its hyperlinks.

Animation movies are for the most part computer generated. There are ways to create graphics that can be adjusted to any size and filled with any kind of color you need. We will learn how to draw a Chess board in Activity 10 – Vector Graphics. You can, of course, change the knight so it looks like your favorite T-Rex dinosaur.

What about the Holy Grail of artificial intelligence? What greater challenge than teaching a computer how to perform something only intelligent human beings were thought to be capable of? Yes, we will program and be able to play Chess against our own program in Activity 11 - Chess. Our brain really does work differently than our Chess machine, so is it fair to oppose a human to a machine?

Capture your programming successes with the Progress Card. You can be proud of your achievements. Every step you complete opens the door to Science a little bit wider…

Conventions

Important parts of the Java programs are explained and presented as follow. Line numbers show up on the left so you can easily look them up inside the whole program you get from the web site at www.funsciencewithyourcomputer.org. The Java code below is taken from Activity 11 - Chess and shows lines 14 to 15 of the Chess.java program. If the current chess game is over then the program will stop playing…

```
14      if (chess.gameOver())
15        break;
```

Each activity is followed by two hands on exercises – similar to two scientific experiments conducted with your computer. You modify the Java program of the current activity and then run it. Four symbols are used to describe each experimental step. They show to the right of the text in the exercises.

The magnifying lens signals a portion of code to search in the Java program. The text left to the lens will specify which lines of code and the actual code will often be shown in the book to facilitate the reading.

Modifying a program is about removing some of the existing code and replacing it with new code. The cross image asks you to remove a portion of code in the Java program. Text to the left will mention the line numbers to be removed e.g. lines 5 to 15.

The plus sign in a circle tells you to add some new code – often after you removed some of the old code. The text right to the plus sign specifies where the code should be added and the actual new code follows after the plus sign.

Run the Java program you are editing in the exercise when you reach the following start button. Text left to the button will describe what you should pay special attention for when the program executes on your machine.

I would like to thank my friends and coworkers at the IBM Ottawa Labs for their feedback and detailed reviews. I am particularly indebted to Jim Des Rivieres, Mike Wilson, Carolyn McLeod, Nick Edgar, Jean-Michel Lemieux, John Arthorne and Adrian Cho. The book is filled with many imperfections only because I did not always include all of their corrections and ideas in the current edition. I wish to thank Professor Martin Moskovits, Dean, Division of Mathematical, Life and Physical Sciences at University of California, Santa Barbara and Professor Pierre-Gilles de Gennes, Nobel laureate in physics. Their working style convinced me the conquest of knowledge is a fine art requiring an extraordinary blend of interdisciplinarity and collaborative skills. That also means *everyone* can enjoy research and engineering activities with mentors showing them the true human face of Science, its many doubts and trials.

It is now time for you to meet three new friends. Sarah, Mike and Liliane will travel with you through the gates of Science. Their guide is a retired professor familiarly referred to as Alex…

Have a good journey!

Chrix

Introduction

Sarah and Mike have arrived at their summer camp located in a small village in the French alps. Sarah is twelve years old and her brother Mike is eighteen. They both come from the province of Ontario in Canada. It is their first time outside of North America and they are very excited about this adventure in Europe. Now they are seated in front of an old chalet on a sunny day in the Alps. The chalet is filled with hiking equipment, but also computers and telescopes. Yes, they really are at a Science camp!

The two teenagers are meeting with Alex and Liliane who supervise their group. Alex takes care of the science and computer activities. He is a retired professor who worked in many different parts of the world. Liliane is helping Alex. She studies literature and history at the university La Sorbonne in Paris. Taking part into a summer camp is a great summer job, she thinks. Alex was repairing an old and dusty computer.

"What an antique! That has to be over fifty years old..." Mike said.

"Alex, what are you doing with that strange box?" Sarah asked.

Alex observed the two teens. He did not say a word and continued to type on the computer keyboard. Liliane was returning from the kitchen, bringing fresh croissants and milk to the table. Soon they were happily eating. Sarah was observing the professor with increasing curiosity and impatience. She asked again. "Alex, really what are you doing?"

Alex looked at the teens again. Sarah had her tiny music player, headphones and cell phone hooked to her belt. Mike was making himself comfortable playing video games on a small portable console.

- "Do you know how your cell phone works, Sarah?" Alex asked.

- "Of course I know." Sarah was feeling a little bit insulted and her brother was shrugging. She took the phone and showed it to Alex. "Press the green button located here. Turn this wheel to select your phone number. Press the wheel again. And when someone calls me, it plays my favorite tune."

- "I didn't ask if you know how to use it, Sarah. Do you know how it remembers phone numbers? Do you know how it plays your favorite song?"

Sarah was a little bit confused by these unusual questions. "Well, it has a computer inside, I suppose."

Alex turned toward Mike. "And you Mike, do you know how are games created on your console?" Mike felt a little bit annoyed. He had the best console, with the best animation. What kind of question was that? He could use a cartridge or download games from the web. And then he could play alone or with friends. Though in this remote village, he had not found anybody to play with. And Alex's antique certainly could not play his favorite game, could it?

Alex turned the screen so they could see it. It was filled with green characters. “See this? Can you guess what it is? It's my first program. I was your age, Sarah and that computer was the first computer at the school in our village. We didn't want to waste time and money on commercial games. So instead I created my own! Here, here is a Chess game.”

Mike and Sarah sat near Alex. What could possibly be done on such an old machine?

“How do I move the pawn? There is no mouse.” Mike said.

“Use the keyboard. Type in the coordinates of the piece and where it should go - e2e4 for example.”

Mike wasn't convinced. This looked really horrible. “There’s no music.” he said. Sarah didn't know how to play Chess so she let Mike try it out. After a few moves, Sarah noticed a message on the screen “Black Wins”. Her brother was looking very frustrated. Alex was smiling.

“How did you do that?” she asked.

“You simply learn how to talk to the machine so you can teach it everything that comes to your mind. It doesn’t come all in one day. We can start with simple programs that do fun things. A program is made of text instructions telling the machine how to do different things. You enter the program into the machine. You don’t have to understand every detail of the program, just copy it from a programming book or get it from the Internet. You’ll see if it makes sense to the machine – it works or it does not work. When it works, modify the text here and there to see how it runs differently on the computer. Over time, it becomes quite natural, like learning how to swim or skate. It takes some practice and someone to guide you through the first steps.”

Mike was looking at the green screen. “So I could also learn how to write a chess game and beat this old chess game?”

“You sure could.” answered Alex. “By the end of summer, you could certainly do that.”

Throughout the summer, Mike, Sarah and Liliane gathered with Alex a few times a week. With Alex's patient explanations, they started to get the feeling that it was really fun to be able to program a computer to do anything you could imagine!

How to run the programs in this book

The programs in this book can be run on personal computers, of the kind you most likely have at home, at school or at work. They are written in a very widely used language called Java.

The Java language offers many advantages. It is free to install – you don't need to spend money buying special software. It is very popular and appreciated at university and in the professional world. So you can be sure time spent studying these programs is bringing real and valuable skills. Finally, most computers can run Java programs.

Today's computers are used to send emails, browse the web and download digital photos from your camera. Programmers need an extra step. They need to install a few tools. Go to our web site www.funsciencewithyourcomputer.org and follow the instructions.

The book contains screenshots showing what the program looks like once executed on the machine. Seeing is not enough. Try running these programs on your computer, by yourself or assisted by a friend or a teacher who knows a little bit about programming. Only then is the experience complete. There is no trick. The machine does what you tell it to do. In a way it is magic, at times frustrating. Programmers can spend hours trying to understand why the computer does not do what they expect it to do. This book wants you to discover this feeling too.

Once you have installed the tool to edit and run Java programs, download the small Java programs used in this book from the www.funsciencewithyourcomputer.org site. After you successfully edit and run a program, take some time to congratulate yourself. Look back at the program. Try to modify it here and there. You don't have to understand everything before running it – nobody really understands everything in general. Experiment and see what the effect is on the computer when you run the program again after making some changes.

The activities are organized to encourage you to further modify the given programs. For example you can modify the graphics of the chess game, or change the values of certain arguments in the program to calculate the number Pi with more or less accuracy. Play. Try things out. Fail, try again, and succeed. That's it. You have become a programmer.

Although programs in this book are written in Java, this is not a book about the Java language. Excellent and in-depth books exist on this topic. It is about programming in general and about turning entertaining ideas into simple programs in particular. It is not about writing perfectly optimized Java programs – the fact is the code in this book will not please expert Java programmers. The

decision was made to write simple programs that are easy to read, using an extremely limited set of language features. All too often, there is a false sense of complexity resulting from the multiplication of choices available to programmers. There are so many ways to solve a given problem - this in itself creates problems. Imagine going to a restaurant with no daily specials. Everything is customizable with dozens of choices for mineral water, etc. In a few rare cases such as on your wedding day, that would be a good thing. In daily situations, deciding what to order takes more time than you can afford during your lunch break. Don't run away from programming and Science because you don't understand 'everything' immediately. You will be in control of the examples given in this book.

Progress Card

You can track every progress you make with the following progress card. Go through an activity. Run the program related to that activity. Praise yourself and mark that activity as completed in table 1 below (indicate the date). Your job title progresses with the number of activities you have completed.

Table 1 - Activities completed - progress

Activity	Completion date	Score	Job Title
1	.. / .. / ..	50	Developer
2	.. / .. / ..	100	
3	.. / .. / ..	150	Consultant
4	.. / .. / ..	200	
5	.. / .. / ..	250	Committer
6	.. / .. / ..	300	
7	.. / .. / ..	350	Senior Developer
8	.. / .. / ..	450	
9	.. / .. / ..	550	Senior Consultant
10	.. / .. / ..	650	
11	.. / .. / ..	1000	Architect

Each activity includes two exercises. An exercise teaches you how to modify the program and improve on a particular aspect of the activity. Once you have completed all the exercises for a particular activity, fill up the corresponding entry in table 2. You have demonstrated that you are now an expert in that field. Scientists frequently gain expertise in two or three fields. Some are famous worldwide because they have knowledge and skills in very important areas that no one else can match.

Table 2 - Exercises completed per activity – expertise

Activity	Exercise 1	Exercise 2	Expertise
1			Probabilities
2			Monte Carlo Simulation
3			Statistics
4			Natural Language Translation
5			Framework design and reflection
6			Graph Theory
7			Performance
8			Classical Physics and Laws of Gravity
9			Web
10			Scalar Vector Graphics
11			Game Theory

The purpose of this book is to encourage you to program by yourself - and with other students or friends. After you complete a couple of activities and exercises, think about programs you would like to create. Modify one of the programs from a completed activity. It is convenient to start from a program that works instead of starting everything from nothing. Give it a name and do something unique with it. Once your own program works well, write its name, the date you completed it and a short description. The more programs you write, the more experience you get. See the extra job title you get in table 3 below based on how many programs you wrote by yourself - or working with another person, which is another very good way to learn.

Table 3 - New programs created by yourself or with a friend

New program	Completion date	Description	Job Title
1.	.. / .. / ..		**Inventor**
2.	.. / .. / ..		
3.	.. / .. / ..		**Chief Inventor**
4.	.. / .. / ..		
5.	.. / .. / ..		**Research Officer**
6.	.. / .. / ..		
7.	.. / .. / ..		**Program Manager**
8.	.. / .. / ..		
9.	.. / .. / ..		**Director of Research and Development**
10.	.. / .. / ..		
11.	.. / .. / ..		**Chief Technical Officer**

The challenge is to become an architect – table 1 – with as many as eleven areas of expertise – table 2 – who works as Chief Technical Officer – table 3.

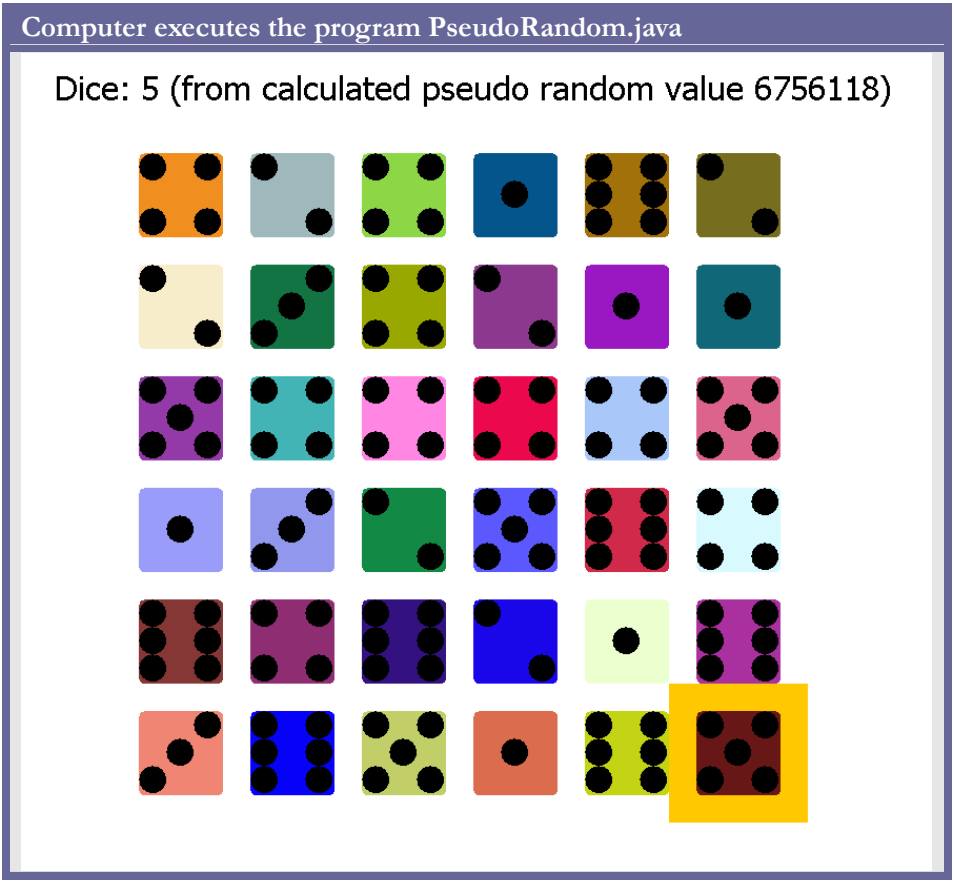
Special note to teachers and parents – course work

Progress cards can structure a course work. Teachers and parents have the difficult task of propagating knowledge from one generation to the next. I hope you will find the activities and exercises helpful in giving colorful faces to Science. Motivated students will challenge and master their materials rather than just strive to survive or skip science classes. Programming in this book plays with Mathematics (geometry, trigonometry and statistics) and Physics (classical mechanics). We show Science in motion with examples taken from recent discoveries. Students can see for themselves that what they learn at school is concrete and necessary to resolve real problems – even fun ones like the ones presented here. Science serves a purpose, and it goes beyond getting the best grades to get admitted into the best university or college. It is also about having fun challenging what we know and can do with our world.

Activity 1 – Pseudo Random

Alex was listening to the results of the 6-49 lottery on the radio. Then he sighed and turned off the radio. He had obviously not won. "Mike, Sarah, come here", Alex said. "How would you generate random numbers with a computer?"

Computer executes the program PseudoRandom.java



Mike and Sarah were certain it ought to be really easy. Alex asked for ways to create random numbers without a computer.

"Throw a dice and you look at the result" said Sarah.

"Yes, a dice is good. It generates a random number between 1 and 6."

"Use the letters in the Scrabble game, put them in a bag and grab one" Mike suggested.

“True. That's good for the Scrabble game, but if you want any letter to have the same chance to come as any other, then you need to make sure you only put one of each letter in the bag, right?”

“Oh I see. Scrabble has rare letters only once, but many times very common letters. So you'd have many more chances to pick those. Yes you're right.”

“If you want numbers between 1 and 49, you can use the results of the lottery.” pointed out Sarah. She had followed the draw on the radio carefully. Imagine if Alex had won!

“Correct again” said Alex.

“You can say numbers as they go through your mind.”

“Do you really think so? Let's verify that one. Mike, write down one hundred times the letters H or T in the order that comes to your mind. Sarah, flip this coin, throw it a hundred times and record the results - heads or tails.”

Mike handed out the two pieces of paper. Alex circled the portions in the list where the letter H was repeated - like the coin had given heads two or three times in a row for example. He also circled the portions where T was repeated. And he compared the two series from Mike and Sarah. "Sarah, this is yours, Mike that's yours.”

Mike and Sarah's curiosity jumped up. "That's right! How did you know?" they asked.

“In Sarah's list, there are many more repetitions - sequences where the same result happened consecutively. That's one indicator of true randomness - there is a significant probability of having the same side of the coin occurring five or six times when you flip the coin a hundred times. When a human tries to produce random series of H or T choices, we tend to think it is not random to give the same result five or six consecutive times - it feels artificial. This is a known psychological behavior which can be used to detect true random versus human random results.”

“That is so weird... So there is random, and random?”

“Yes. Random has its laws too. They can be applied to find people who lie about their taxes.”

“How is that possible?”

“People who make up numbers about how much income they declare in different parts of their business often do a very poor job at bringing up random numbers or numbers that have the same statistical properties from data originating from real taxes. When the government detects such deviation, they target these individuals for further inquiries.”

The teenagers were a little bit shaken. They had thought something was either random or not. And now they were starting to realize things were a little bit more varied than they had imagined.

“So now, how would you generate random numbers with a computer?” Alex asked again. “Think about it for a while. It's time for my national news on TV.”

Half an hour later, Alex came back. “How did it go?” he enquired.

"Well, we thought it would be easy. The computer knows how to calculate, so we thought of a formula to generate random numbers." Sarah explained.

"We have defined a very complicated formula. It's pretty smart. It's doing a bunch of cosine, sinus and logarithmic computations based on what digits are in different positions in the number." proudly presented Mike.

Alex looked at it. "And does it work?"

They showed him some numbers the program had generated. The numbers looked quite different.

"What initial number did you use?"

"We just picked one with different digits."

"And what happens if you restart the program?"

Of course, the exact same series of numbers were reproduced. "Do you see where I'm getting at?" asked Alex. "Your numbers look different. Yet, the next number is entirely defined from the previous number. So if two machines run your program, starting with the same number, they print the same results from there."

"A random number has no memory of the numbers that come before it. When you flip a coin, it does not matter if just before the coin had been flipped on tail or on face. If you throw a dice, it does not matter if you previously had a pair of six. You are still just as likely to get a one as you are from getting any of the other sides. So your program does not generate random numbers."

Mike and Sarah were disappointed. But they agreed with Alex. Yes, Alex was right.

"So how do you do it, Alex? What is the solution?"

Alex frowned, and smiled at them.

"You can't. A computer cannot generate random numbers." And he waited for their reaction.

Now the teenagers did not want to believe him. He had to explain.

"You are very close to it. What your program does is generate pseudo random numbers. Numbers that have some statistical properties of true random numbers. Yet there is a hidden order in your result which you can use to predict the next number. It's determined by the very mathematical formula that you picked. It will always yield the same result for a given input."

"That's exactly what computers were made for. To calculate and be reliable, so you can count your money, count votes, make simulations and always get the same results for a given set of data. Imagine you go to your bank to deposit a one hundred dollar check and sometimes the bank adds eighty dollars, sometimes it adds one hundred and five dollars (you could see Alex was not very kind with his banker as he assumed the banker's computer made mistakes that profited to the bank, not Alex...). That would be a difficult world to live in, isn't it?"

"The good news is that for many uses, pseudo random numbers are just as useful as true random numbers. And they are used extensively for many

simulations, games. Many mathematicians and programmers work very hard to bring up increasingly reliable pseudo random number generators."

Liliane's notes

Alex's program uses a pseudo-random number generator invented by a person who is very famous for his work on computers last century. John Von Neumann invented that method to produce apparently random numbers in 1946. He knew they were not truly random. He knew you could predict the next number based on what current number was used. He knew the numbers repeated themselves after a certain period. But the method was simple and it could be run by the computer very efficiently - much better than what Mike and Sarah had originally invented.

Take a number with eight digits. Multiply this number by itself. You get a number of sixteen digits. Remove the first four and last four digits and you now have a new eight digits number. Repeat as many times as you need eight digit numbers. You are generating what looks like a series of random eight digit numbers.

How do we select the initial eight digits number? That is a very good question indeed. If we run the method starting from the same initial eight digits number, then we obviously get the exact same series of numbers. So they are not random. They are pseudo random. An important step is to randomly select the initial eight digits number so that we don't run the same results every time the program is started. And it looks like we are pretty much forced into a corner here. That's precisely what we were trying to solve initially... Chicken and egg story. In practice, a program can use the amount of milliseconds in the computer's clock. It is unlikely that running the same program on two machines or on the same machine at different times will happen at the exact same millisecond. So for many applications, we can live with that kind of technique.

There is one more thing to perform so we can truly use our pseudo random numbers. We have eight digit numbers. But what if we want to use our program to replace a dice? So we want numbers between one and six. We could extract the last digit in our eight digit number but that would be a number between 0 and 9. There are different ways to achieve this. A common technique is to remove six from our eight digit number. And repeat again as many times is needed, till the resulting number is lower than six - i.e. between zero and five. This is so common of an operation that all computers actually know how to calculate it very quickly. It's called modulo. Its symbol is %. For example 13 % 6 is read thirteen modulo six. That's equal to 1. 13 – 6 equal 7 which is still greater than 6. 13 – 6 – 6 equals 1 which is now lower than 6. So we get our eight digit number modulo six, and we obtain a number between zero and five. We want a number between one and six. We simply add one and we are done. We now can simulate a dice using our eight digit pseudo random number generator.

Here is how to actually teach the computer how to do it.

This defines the name of our program to create and display pseudo random numbers. Giving a name to our program allows other programs to use it (that's why we say it is 'public'. We can choose any name, so it's best to pick one that reminds us of what the program is for.

```
9 | public class PseudoRandom {
```

Our program starts here. This is the main part of the program that gets first executed by the computer.

```
11 |   public static void main(String[] args) throws
   | Exception {
```

First we initialize our program with a number. Here we chose the number 12345678.

```
12 | PseudoRandom random = new PseudoRandom();
13 | long value = 12345678;
```

We request a portion of the computer's screen so we can draw images of dices later.

```
14 |    Screen screen = new Screen();
15 |    screen.setVisible(true);
```

We want the program to create and display random numbers. So we tell it to repeat the instructions that follow this line for ever.

```
16 |    while (true) {
```

We ask the computer to give us a pseudo random number using the previous pseudo random number.

```
17 |      value = random.calculateNext(value);
```

What we really want is simulating a dice. We need to transform our pseudo random number so that its value is between one and six. That's taking care of by our dice function.

```
18 |      int dice = random.dice(value);
```

We now simply need to ask the computer to display it on the screen. For fun, this will draw an actual dice, not just a plain number. And the color of the dice will be based on the value itself (screen can show millions of different colors, it will use the one that corresponds to our pseudo random number).

```
19 |      screen.showDice(dice, value);
```

We pause the program for one second (one thousand milliseconds) so we have some time to read the screen. If this line is removed, things would go really fast.

```
20 |      Thread.sleep(1000);
```

We tell the computer where the part of the program that must be repeated indefinitely ends. And then we say we have also reached the end of the main part of the program.

```
21 |    }
22 |  }
```

The main method asks the computer to create and display pseudo random numbers. It does not actually teach the computer how to create a pseudo random number. That is done elsewhere in the program in the function named caculateNext.

This function uses a mathematical formula to create a different value from the given value. It returns that new value. That new value looks apparently very different and unrelated to the previous value – it looks random.

```
24 |  public long calculateNext(long value) {
```

That's how the next value is calculated – by multiplying the previous value by itself.

```
25 | value = value * value;
```

And the function ends by returning the new value it has just calculated.

```
31 |    return value;
32 |  }
```

We calculated a large number that looks random. We are going to use some Mathematics to turn it into a small number that is between one and six. We create the function named dice. Given our large value the function returns a

small number between one and six. As we have seen earlier, 'value % 6' is read 'value modulo six'. That looks complicated but all it does is removing six to the value as many times till the result is lower than six – between zero and five. After that we add one, and voila.

```
34 |  public int dice(long value) {
35 |    return (int)(value % 6) + 1;
36 |  }
```

We have simulated a dice that has six faces – one to six dots.

See also the following related activities

- Monte Carlo
- Random Walk

To go further

- 1654 - Blaise Pascal and Pierre de Fermat study gambling problems and develop the mathematical theory of probabilities. Prior to their work, most people played dice and card games without really being aware of the odds of winning or losing.
- 1687 – Isaac Newton publishes the foundations of classical mechanics. The world is mechanical and as long as one knows accurately its state at a given moment, one can predict any future state. There is no room for true random behavior.
- 1925 - Heisenberg and Schrödinger develop the foundation of modern quantum mechanics, along with Albert Einstein and Niels Bohr. Unlike what Mr. Einstein would like to believe in, it now appears that God does play dice with the universe. Certain natural phenomenas are truly random such as the emission of neutrons by radioactive materials. The emission of a neutron at a particular time from the radioactive source can happen at any interval of time independently of when was the last time the source emitted a neutron. Yet at large scales, the average emission of neutrons can be measured. But at the scale of a single neutron, it is completely random. Such fundamental random phenomenas are now used to manufacture random number generators. These devices can be connected to a computer so that a program can ask the device for true random numbers.
- 1946 - John Von Neumann develops the Middle Square method to rapidly produce series of pseudo random numbers on the ENIAC computer (to run Monte Carlo simulations). Mr. Neumann is a

mathematician who contributed to the development of the atomic and hydrogen bombs at Los Alamos in the United States.

Exercise 1: Create different pseudo random numbers every time the program is executed

Run the program twice. Notice each time the same series of numbers are displayed. To remedy to that, replace the line

```
13 | long value = 12345678;
```

With that line

```
13 | long value = System.currentTimeMillis() % 100000000;
```

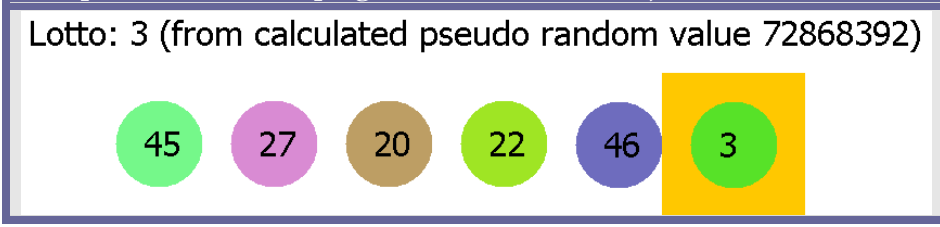
Run the program twice again. What do you observe?

Now the computer is getting the exact current time (given in milliseconds) at the moment that part of the program is executed. It computes the first pseudo random number with eight digits. Every time you run the program the time is a little bit different so that number is also different. Each time you get a series of pseudo random numbers that look different from the previous execution.

Exercise 2: Modify the program to simulate Lotto 6/49

Modify the program so that it calculates and displays six numbers between one and forty nine.

Computer executes the program PseudoRandom2.java



First edit the function dice.

```
34 |   public int dice(long value) {
35 |     return (int)(value % 6) + 1;
36 |   }
```

Replace the number 6 by the number 49.

```
34 |   public int dice(long value) {
```

```
35      return (int)(value % 49) + 1;
36    }
```

Second, edit the function paint. Replace the line below

```
79          drawDice(g2, i, j, dice, value,
isCurrentDice);
```

With the following line.

```
79          drawLotto(g2, i, j, dice, value,
isCurrentDice);
```

Find the last three lines of the function drawDice.

```
116         g.fillArc(x + 2 * r, y + r, r, r, 0, 360);
117       }
118     }
```

Add the function drawLotto after the function drawDice.

```
119
120     void drawLotto(Graphics g, int i, int j, int
    dice, long value,
121         boolean isCurrentDice) {
122       int width = 80, height = width, r = width / 3;
123       int x = (width + r) * i + width;
124       int y = (height + r) * j + height;
125       g.setFont(new Font("Tahoma", Font.PLAIN, 32));
126       if (isCurrentDice) {
127         g.drawString("Lotto: " + dice + " (from
    calculated pseudo random value "
128             + value + ")", 0, 30);
129         g.setColor(Color.ORANGE);
130         g.fillRect(x - r, y - r, width + 2 * r, height
    + 2 * r);
131       }
132       g.setColor(new Color((int) value));
133       g.fillOval(x, y, width, height);
134       g.setColor(Color.BLACK);
135       g.drawString("" + dice, x + r, y + 2 * r);
136     }
```

Run the program. It should display series of six numbers between one and forty nine. If you have completed Exercise 1, you will get different Lotto results every time you run the program.