INTPROG JAVA INDIVIDUAL COURSEWORK 2015/16

Set : Feb 16thth 2015

Demonstrations: In your practical class March 16th, 17th, 21st

Electronic Submission: Submit zipped Java source files to Moodle by 23.55 on Monday 14th March.

Worth: 20% of unit marks.

Work to be done individually. As the assessment will primarily be a demonstration of your final program it will not be marked anonymously.

AIMS

To develop object oriented programming skills, including designing, documenting, testing and debugging of Java code.

DESCRIPTION

Your task is to write a program which models an imagined universe which has some similarities to the real Universe but also significant differences. In particular, the imagined Universe is a finite, two-dimensional, rectangle. It contains a variety of different objects (comets, stars, planets and black holes) which have different characteristics, the details of which are given below.

To help you get started the BlueJ project Universe has been produced which provides some of the functionality that the final program will need, but much needs adding. You first need to study this project and understand how the code in the Universe and Space_Object class work.

Objects in the Imagined Universe

General

All objects are circular and have a current location (x and y co-ordinates), a size (diameter), a colour (possible colours vary depending on the object, see below), a lifetime and a speed of travel in both the x and y dimensions (static objects have a speed of 0 for both). The final program will have a main loop that processes each object in the universe each time round the loop. An objects lifetime determines how many iterations around this loop the object will exist for. On each iteration the value of lifetime is reduced by 1. When the lifetime reaches zero the object is destroyed and is no longer a part of the universe. The following kinds of object can be part of the universe.

Comets

Comets move through the Universe in straight lines and when they hit an edge, they bounce off the edge. Comets may be coloured red, blue, green, cyan, gray or pink. There are a number of different types of comets that have different behaviours as detailed below.

Stars

Stars are static – they don't move. They are always yellow. They may have one or more planets orbiting them.

Planets

Planets orbit stars in circular orbits and therefore maintain the same distance from their star's centre. Different planets can orbit a star at different speeds which are NOT dependant on their size or distance from the star. You, as creator of the universe, will make sure you don't create planets that collide with the edge of the universe.

Black Holes

Black holes are static – they don't move. They are always black. Strangely, planets and stars are unaffected by black holes but any comet that gets too close to a black hole is sucked in until it

reaches the centre and disappears. “Too close” is defined to be anywhere within a distance from the centre that you choose. Immediately a comet is too close it will change direction to take the shortest route to the centre of the black hole. The comet increases speed at each iteration as it moves closer to the centre of the black hole and all other behaviours (i.e. changing properties, collisions) no longer occur.

Types of Comet

Comets change some of their properties when they collide with an edge of the Universe and also as they are moving. The probability of a change occurring for any iteration is provided when a comet is created. The five things that may change are:

Colour – the comet can change colour to another valid colour.

Size – the radius of the comet can change between a given minimum and maximum radius.

Speed – the speed of the comet can change between a given minimum and maximum.

Direction – comets normally bounce off the edge of the universe at the same angle they hit it at. Comets that change their direction do not bounce off at the expected angle.

Lifetime – the lifetime of a comet is reduced.

The table below lists 10 types of comet and their behaviour (what property changes) both when they hit an edge and for a given % of times when they move.

	Edge	Moving
Comet0	Colour	Size
Comet1	Size	Speed
Comet2	Speed	Direction
Comet3	Direction	Lifetime
Comet4	Lifetime	Colour
Comet5	Colour	Direction
Comet6	Size	Lifetime
Comet7	Speed	Colour
Comet8	Direction	Size
Comet9	Lifetime	Speed

What types of comet do you have to implement?

You have to write code for 2 different comet types. Firstly the comet type that corresponds to the final digit of your student number. You also choose one other comet type that changes two different properties from your first comet type. For example if your last digit is 4 you will implement Comet4 and one of Comet1, Comet2 or Comet8. You would not be allowed to implement Comet3, Comet5, Comet6,Comet7 or Comet9

Handling Collisions between Objects

With comets and planets moving in the Universe, as well as the static stars, collisions between these objects are certain to occur on occasions. You, as creator of the Universe, will ensure that no stars or planets can collide with each other so all collisions will involve at least one comet.

Any comet that collides with a star is destroyed instantly and provides more energy to the star. The lifetime of the star is increased by the area of the comet destroyed (to the nearest integer).

Any comet that collides with a planet is also destroyed and adds mass to the planet. The size (in area) of the planet is increased by the size (in area) of the comet.

Comets of the same type which collide with each other just bounce of each other with their direction of travel reversed.

When two comets of different types collide the larger of the two continues at the same speed and direction and the smaller is destroyed. If the comets are the same size, the slower comet is destroyed. If both comets have identical size and speed then they are both destroyed.

Functionality 1 (up to 20% of marks)

Create a Universe that has stars, planets and the two types of comet you are using (see above) with their correct behaviours when hitting an edge and moving BUT not including collision detection.

Functionality 2 (up to 25% of marks)

Add the collision detection and actions.

Optional Functionality 3 (additional task up to 10% of marks)

Add black holes

Optional Functionality 4 (additional task up to 10% of marks)

Add a user interface which allows a user of the program to interactively select different universes to run and see that changes to stars and planets, that should occur when collisions take place, are happening. For full marks for the interface you should provide a way to save to file the current state of the universe and subsequently read it back.

6 Submission and Assessment

Please submit via Moodle a zipped file containing all relevant Java source code by 23.55hrs on Monday March 14th 2016. The primary assessment of this work will be via a demonstration, to a member of staff, given in a Java lab class between Wednesday March 16th and Monday March 21st. Each demonstration should last approx. 5 minutes. Failure to demonstrate your program will result in a functionality mark of 0%. Late submissions and demonstrations will be capped at 40% (subject to ECF amendment).

Marks Breakdown

Functionality 1	up to 20%
Functionality 2	up to 25%
Optional Functionality 3	up to 10%
Optional Functionality 4	up to 10%
Testing	up to 10%
Code quality	up to 15%
Documentation	up to 10%

Advice on completing the coursework

You may ask me or your tutor for any help you may require – you will not be penalized for asking for help. We will not do the coursework for you but will answer any specific problems you have. You may also get help from other students taking the unit but if any submissions have an excessive amount of duplicate code the University's plagiarism procedures will be enforced. In simple terms – what you submit must be your work not copied from, or done by, someone else.

All code must be professionally written (e.g. with JavaDoc comments and correct indentation!) and will be assessed for:

- correctness
- appropriate use of language constructs

- style (commenting, indentation, etc.)

Develop your code in small parts. Test each part as you implement it. Compile your program frequently – don't add more than a handful of program statements without compiling it to check what you've done. Keep backup copies of working code.

Think carefully about the design of each class you create – what attributes and services should a class offer? Write methods to test each class you write.

Don't reject well thought out code just because you get compilation errors – work out why you've got the error and correct it, don't throw away the good with the bad.

Don't be over ambitious. Do not spend excessive time on this coursework. You should spend approximately 20-30 hours (3-4 full days).

The coursework brief has deliberately not been entirely prescriptive about exactly what you do – there is room for individual interpretation. If you have any doubt about what is required of you or believe there is any ambiguity in the coursework brief please email me with your query and I will post answers to all such queries on Moodle.

Robert Topp Feb 2016

KEY POINTS

- Each student must create their correct comet types as described in this coursework brief.
- You need to start work on this now!
- I expect you to write tests for each class, some JUnit and some of your own, as appropriate.
- You will attract additional marks for code quality if you correctly create and use an inheritance hierarchy in your solution.
- Attributes of classes can never be public. Only constructors and methods may be public. Your assignment will be capped at 40% if you don't follow this rule.
- Providing appropriate Javadoc comments in your program is all that is needed to get marks for documentation.
- To get a good mark you will need to use arraylists or arrays to hold the objects in the Universe.
- Failure to attend the demonstration will result in 0 marks for functionality.
- Plan your demonstration in advance so you can demonstrate to the marker all your program functionality in the allocated time (approx. 5 minutes).