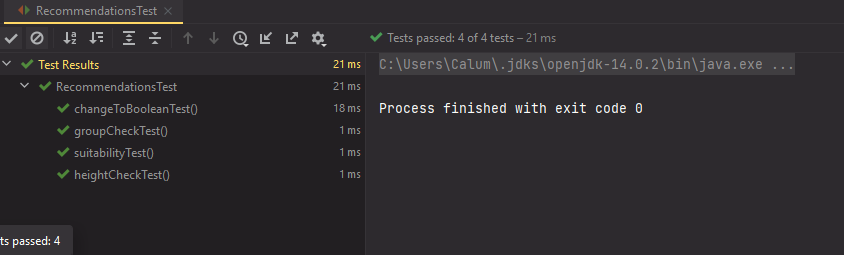
# Testing & Evidence – Step1

## Junit

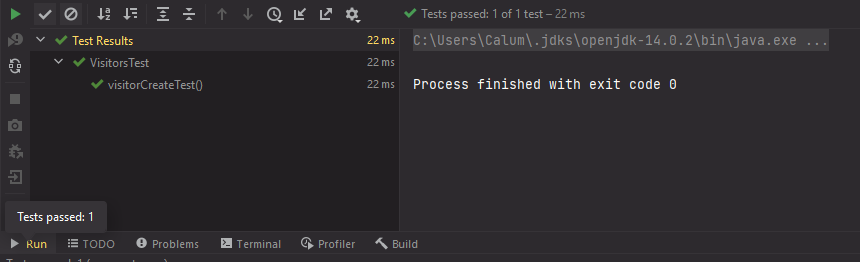
### RecommendationsTest

*/\*  
 \* CS251 Programming  
 \* Year 2, term 3  
 \*  
 \* Coursework Project 2020/21  
 \* by nfb19202 - Calum Doughty  
 \*  
 \*/*import *org.junit.jupiter.api.Test*;  
  
import static *org.junit.jupiter.api.Assertions*.\*;  
  
class *RecommendationsTest* {  
 *Recommendations* reco = new Recommendations();  
 *Ride* ride = new Ride();  
 */\*  
 name = "Testing";  
 typeOfHeight = 1; ">"  
 anotherTypeOfHeight = 0;  
 heightMax = 3;  
 heightMin = 2;  
 wheelchair = true;  
 groupMax = 4;  
 groupMin = 2;  
 theme = 1;  
 adrenaline = true;  
 horror = true;  
 kids = true;  
 water = true;  
 \*/  
  
 @Test* void suitabilityTest() {  
 }  
  
 *//check that this method changes String values to boolean  
 @Test* void changeToBooleanTest(){  
  
 boolean holder = reco.changeToBoolean("Y");  
 *assertTrue*(holder);  
  
 holder = reco.changeToBoolean("N");  
 *assertFalse*(holder);  
 }  
  
 *//check to see if the group check functionality works  
 @Test* void groupCheckTest(){  
  
 boolean holder = reco.*GroupCheck*(ride, 4);  
 *assertTrue*(holder);  
  
 holder = reco.*GroupCheck*(ride, 7);  
 *assertFalse*(holder);  
 }  
  
 *//check to see if the height check functionality works  
 @Test* void heightCheckTest(){  
  
 boolean holder = reco.*HeightCheck*(2.7, ride);  
 *assertTrue*(holder);  
  
 holder = reco.*HeightCheck*(1.7, ride);  
 *assertFalse*(holder);  
 }  
  
}



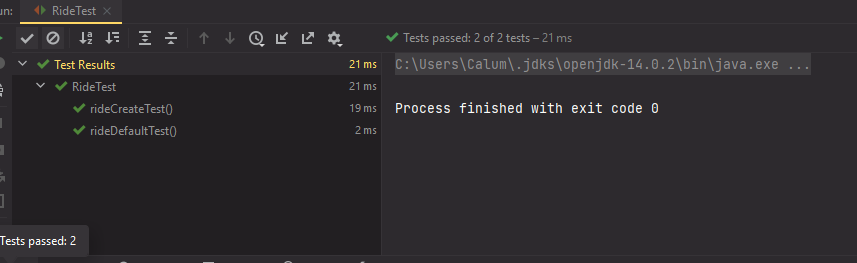
### VisitorTest

*/\*  
 \* CS251 Programming  
 \* Year 2, term 3  
 \*  
 \* Coursework Project 2020/21  
 \* by nfb19202 - Calum Doughty  
 \*  
 \*/*import *org.junit.jupiter.api.Test*;  
  
import static *org.junit.jupiter.api.Assertions*.\*;  
  
class *VisitorsTest* {  
  
 *@Test* void visitorCreateTest(){  
 *//Visitors(double height, boolean child, boolean heightPass, Boolean wheelchairQ, Boolean kidsQ, Boolean waterQ, Boolean horrorQ, Boolean adrenalineQ)  
 Visitors* visitor = new Visitors(1.3, true, false, true, true, true, true, true);  
  
 *assertEquals*(visitor.getHeight(), 1.3);  
 *assertTrue*(visitor.isChild());  
 visitor.setChild(false);  
 *assertFalse*(visitor.isChild());  
 }  
  
}



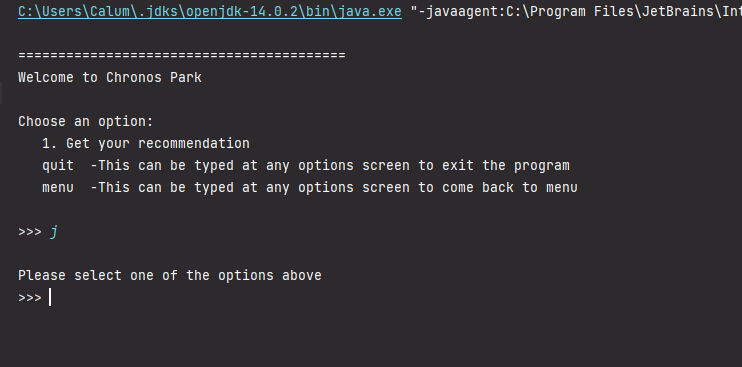
### RideTest

*/\*  
 \* CS251 Programming  
 \* Year 2, term 3  
 \*  
 \* Coursework Project 2020/21  
 \* by nfb19202 - Calum Doughty  
 \*  
 \*/*import *org.junit.jupiter.api.Test*;  
  
import static *org.junit.jupiter.api.Assertions*.\*;  
  
class *RideTest* {  
  
 *@Test* void rideDefaultTest(){  
 *//Ride(String name, int typeOfHeight, int anotherTypeOfHeight, double heightMax, double heightMin, boolean wheelchair, int groupMax, int groupMin, int theme, boolean adrenaline, boolean horror, boolean kids, boolean water)  
 Ride* ride = new Ride();  
  
 *assertEquals*(ride.getName(), "Testing");  
 *assertNotEquals*(ride.getTheme(), 2); *//actual = 1* }  
  
 *@Test* void rideCreateTest(){  
 *//Ride(String name, int typeOfHeight, int anotherTypeOfHeight, double heightMax, double heightMin, boolean wheelchair, int groupMax, int groupMin, int theme, boolean adrenaline, boolean horror, boolean kids, boolean water)  
 Ride* ride = new Ride("Tester", 1, 0, 0, 0, true, 0, 0, 1, false, true, false, true);  
  
 *assertEquals*(ride.getHeightMin(), 0);  
  
 }  
}

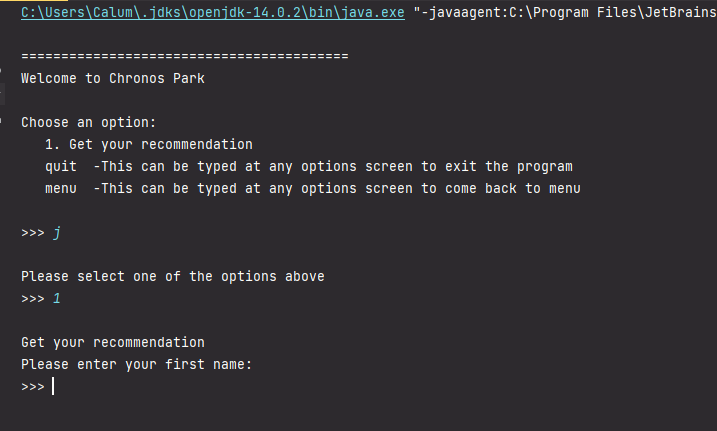


## Data outputs

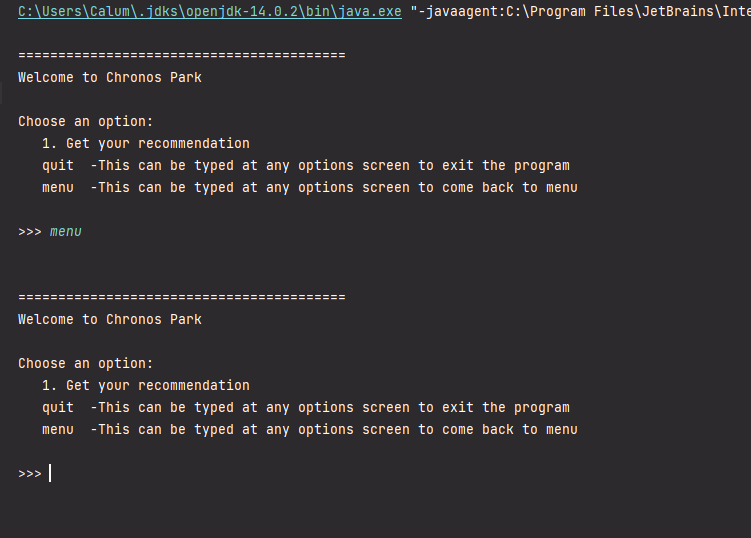
false menu input



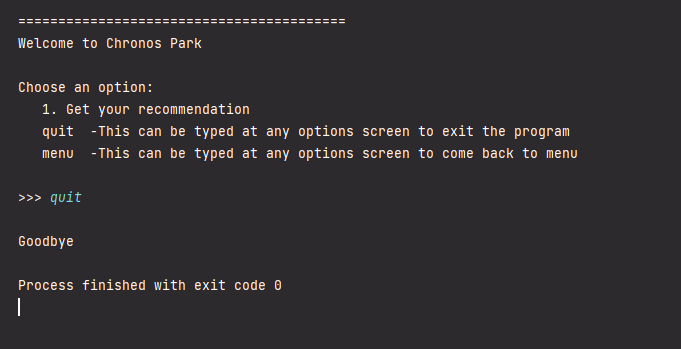
correct menu input



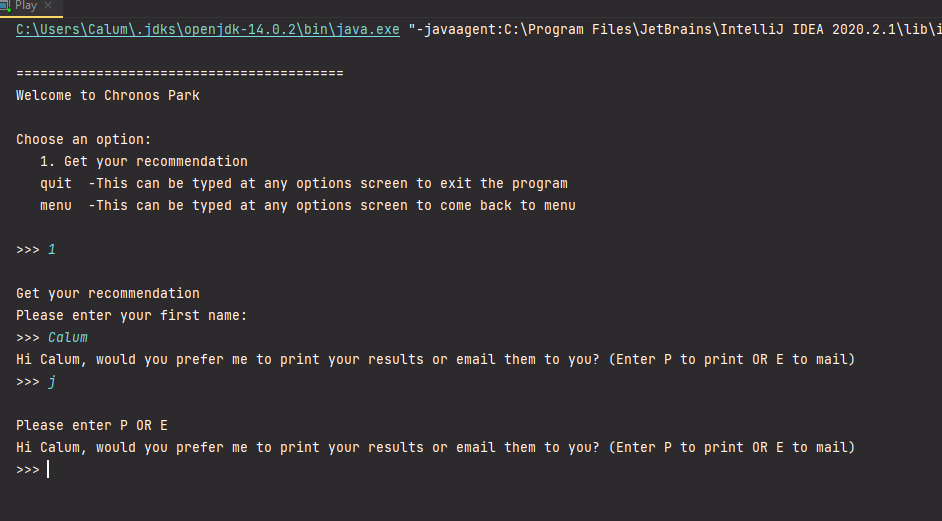
menu input



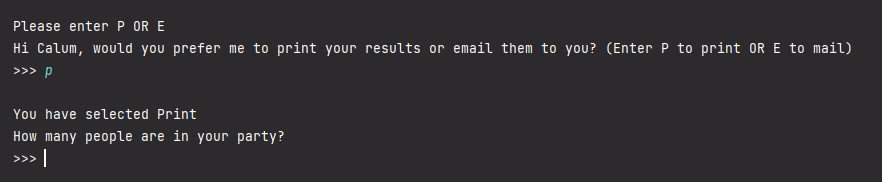
quit input



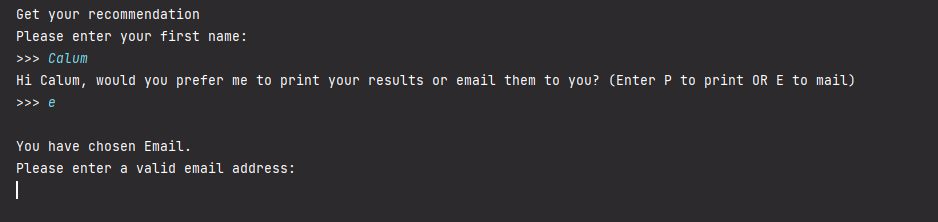
false email selection



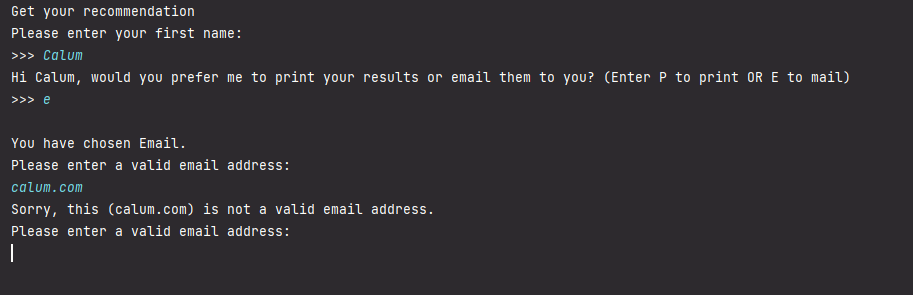
correct print selection



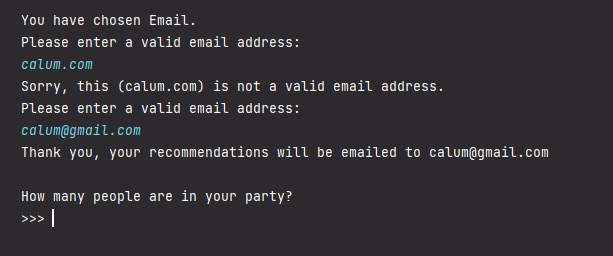
correct email selection



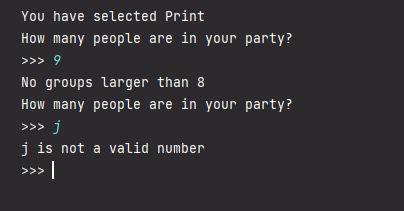
incorrect email input



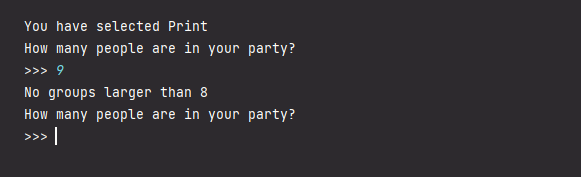
correct email input



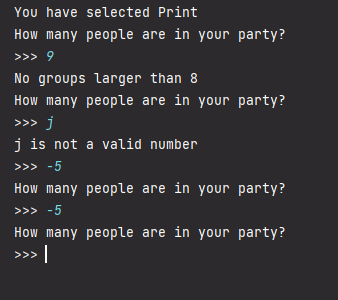
false party No input



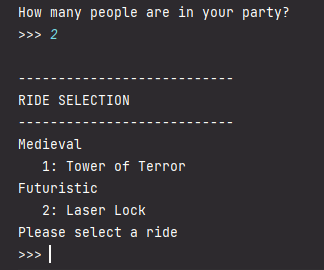
out of bounds party No input



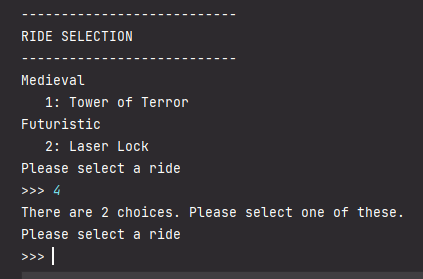
other out of bounds party No input



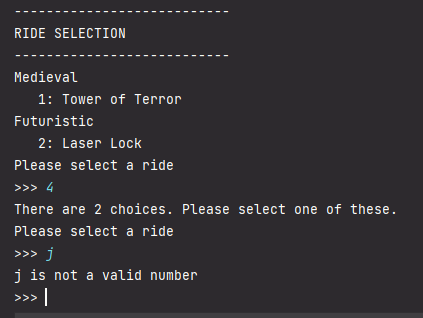
correct party No input



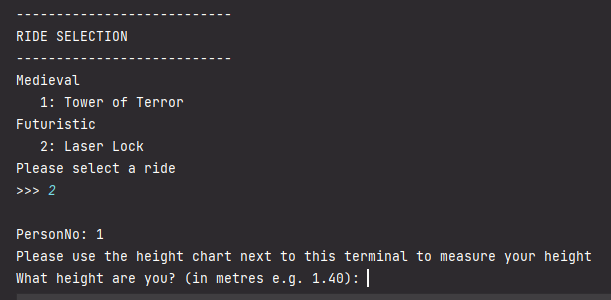
false input ride selection



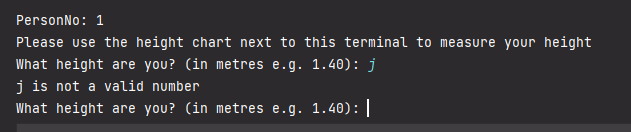
character input ride selection



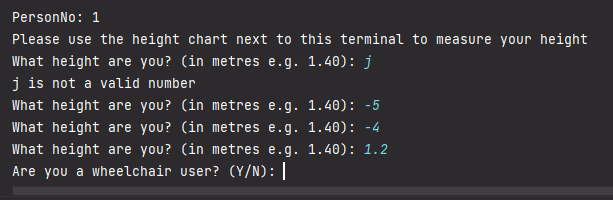
correct input ride selection



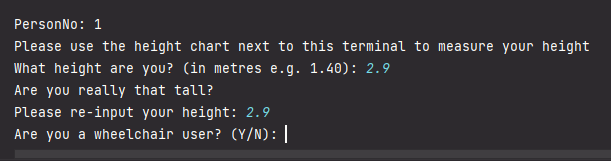
false input height



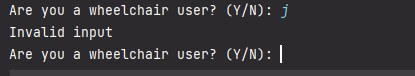
out of bounds height check



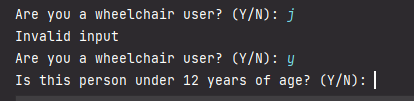
tall person double check



wheelchairQ false input

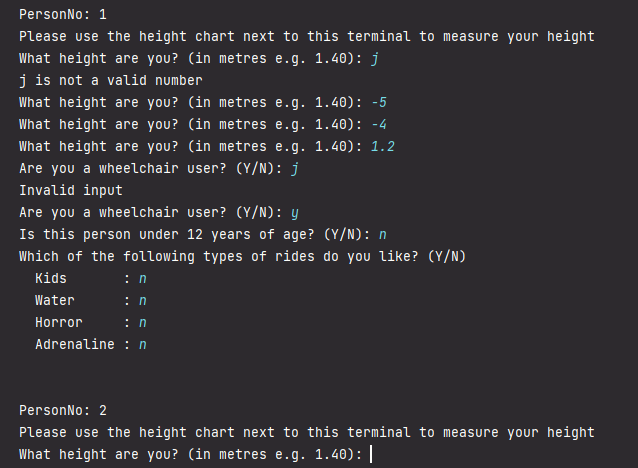


wheelchairQ correct input

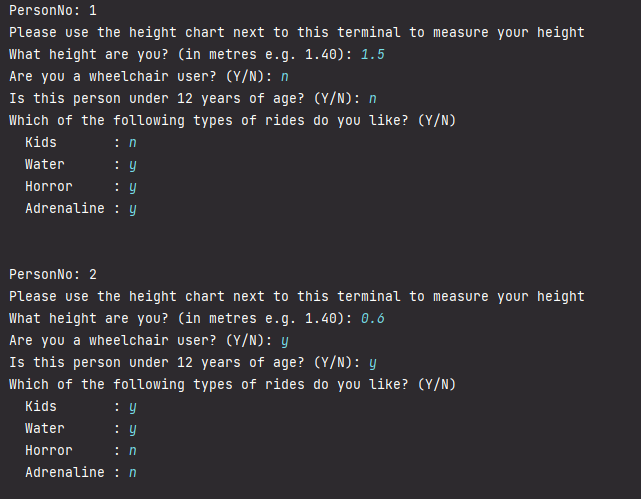


^^^ This type of error handling is identical for all following preference questions ^^^

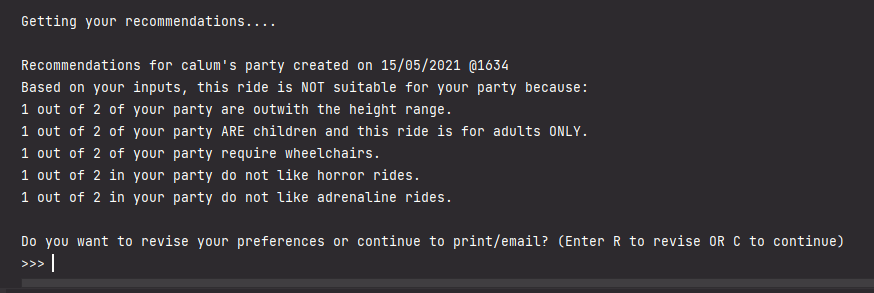
correct input preference questions



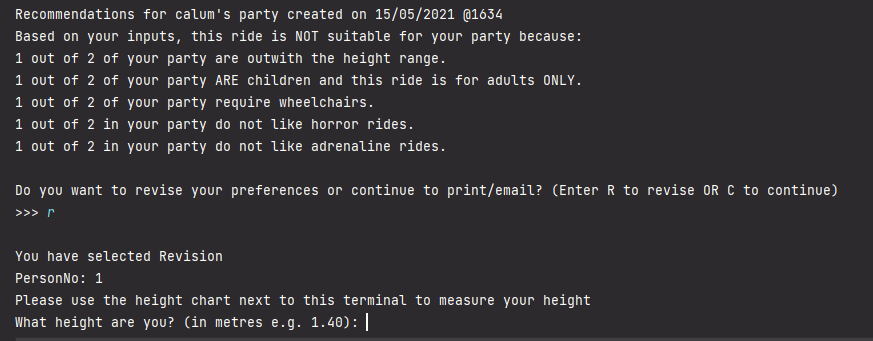




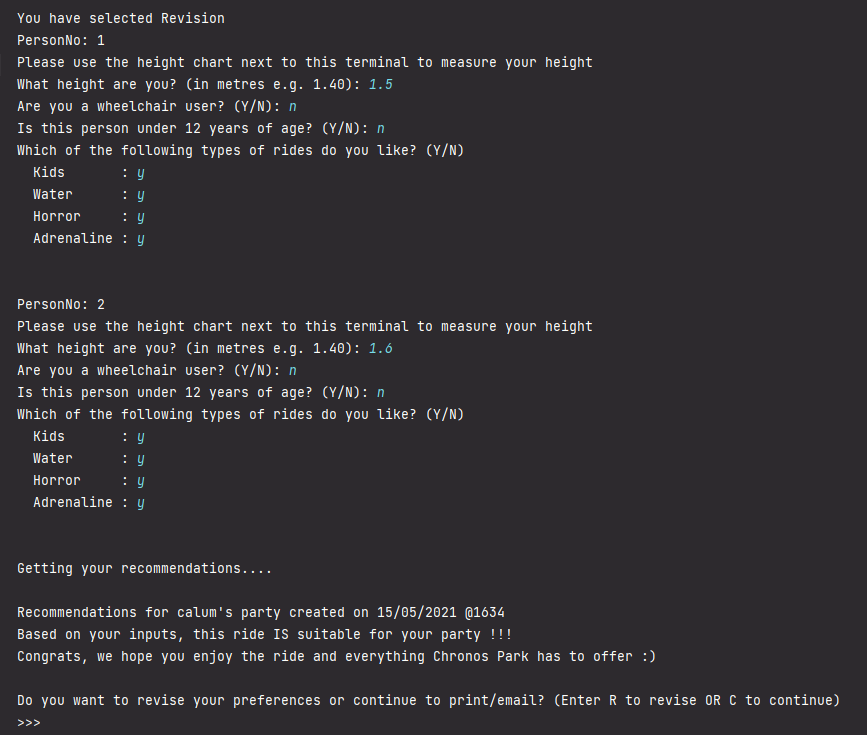
We can see here that this is correct output for input choices



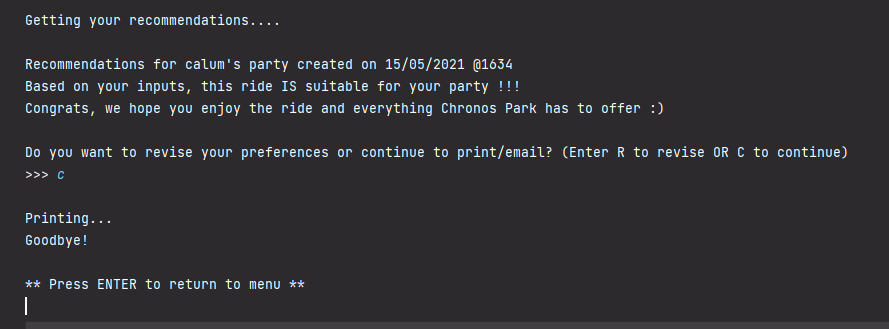
select revision



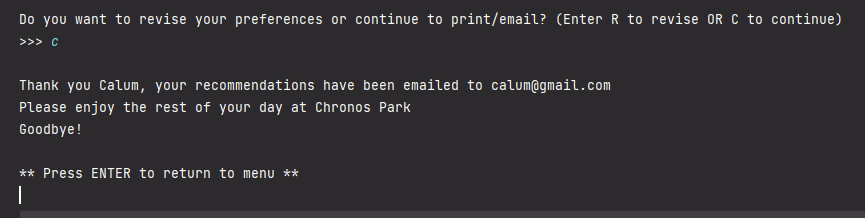
correct output for given input



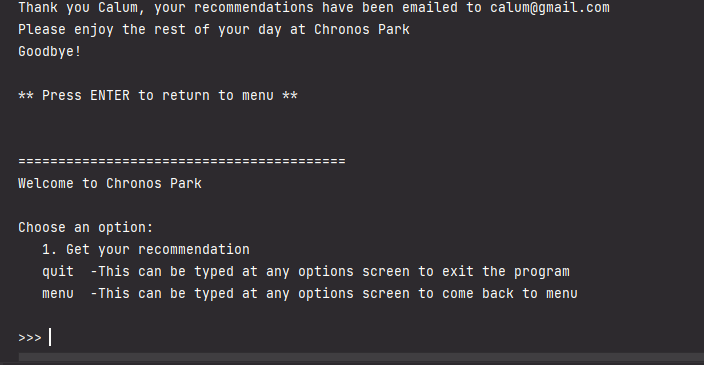
c to continue print



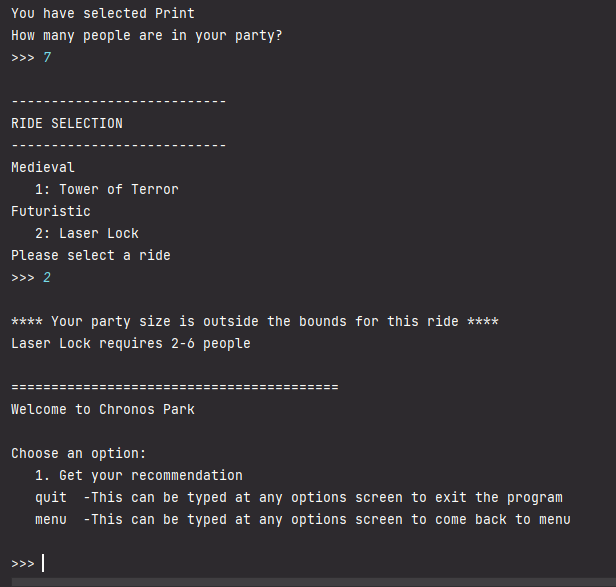
c to continue mail



press Enter



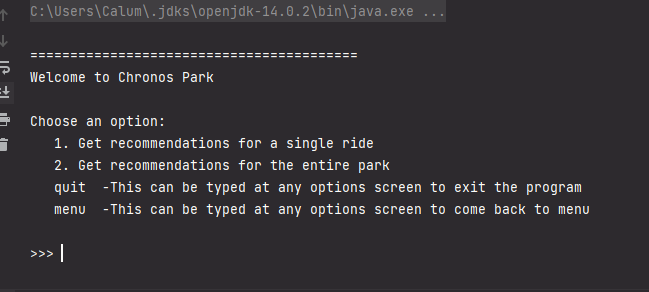
group out with bounds early bounce back



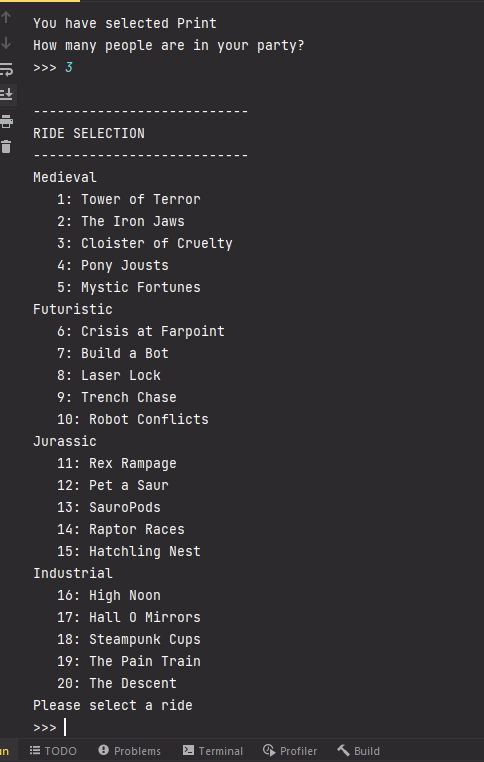
# Testing & Evidence – Step2

## Data outputs

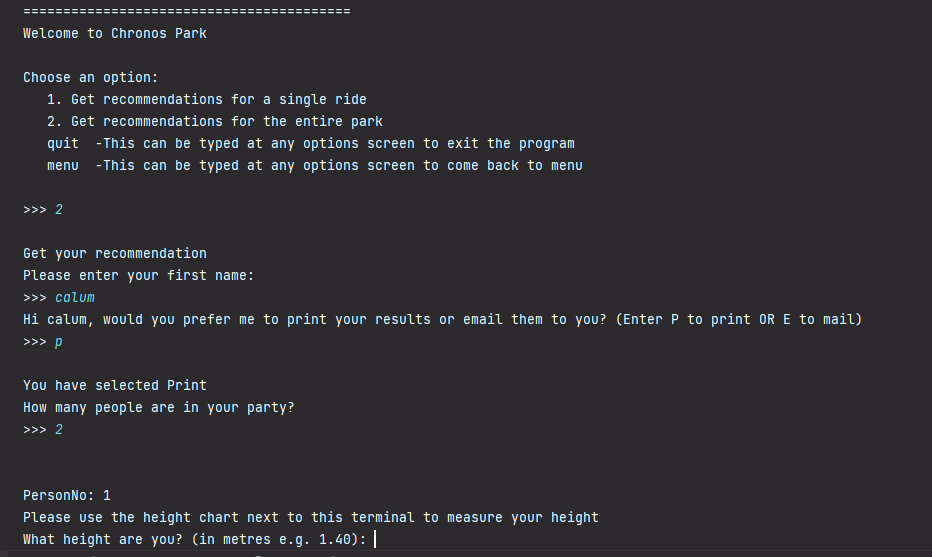
menu



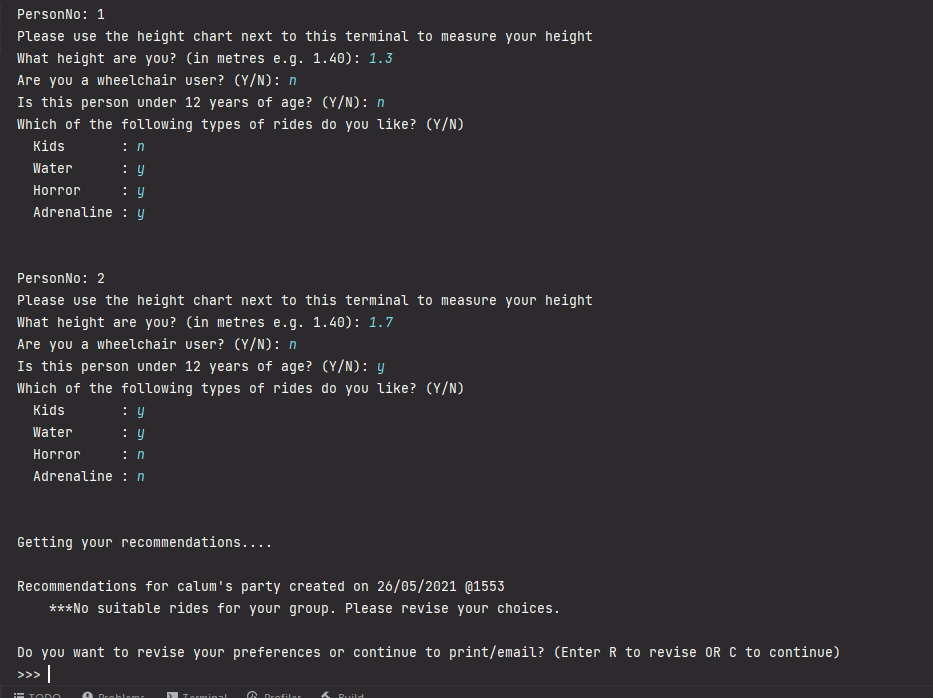
ride selection for OPTION 1 revised



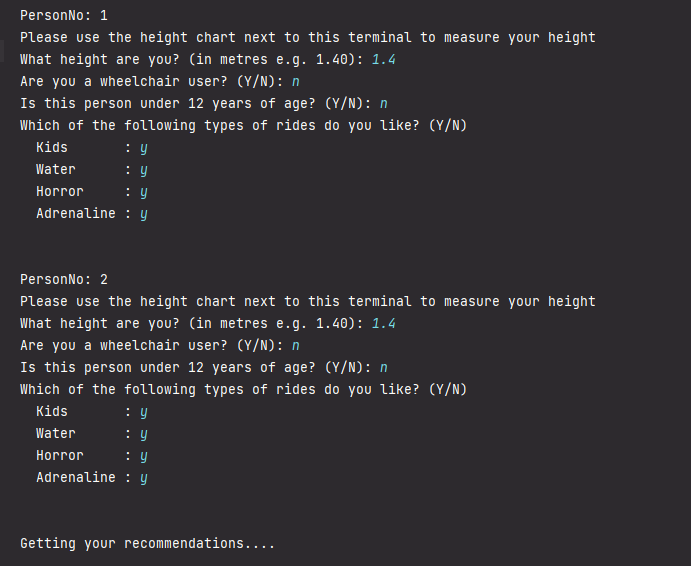
OPTION 2 selection



if rides aren’t compatible



if rides are compatible

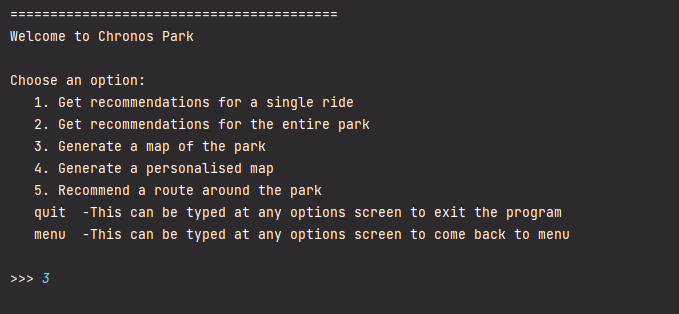




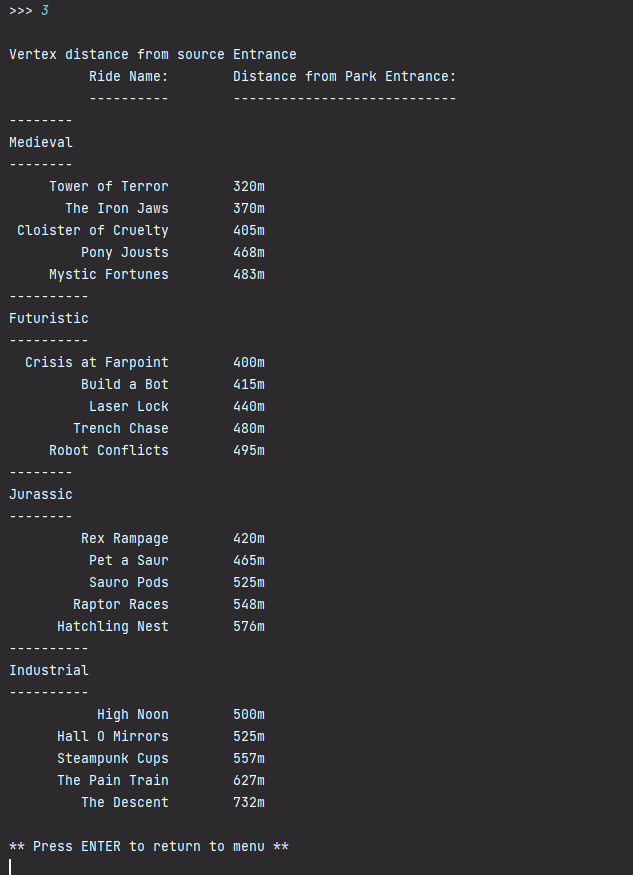
# Testing & Evidence – Step3

## Data outputs

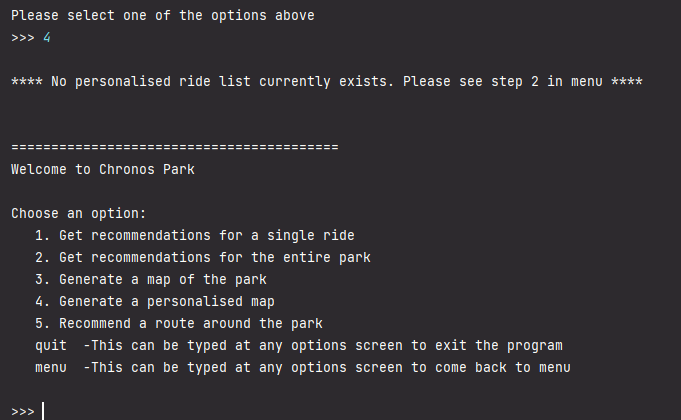
menu



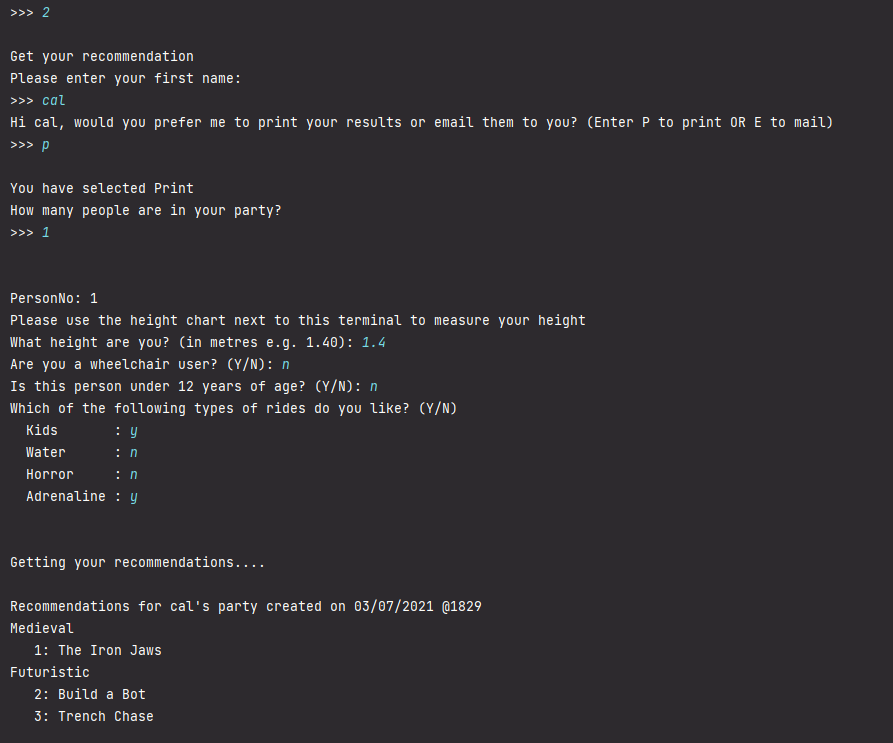
map of the park



no personalised input (part 2 yet)



testdata



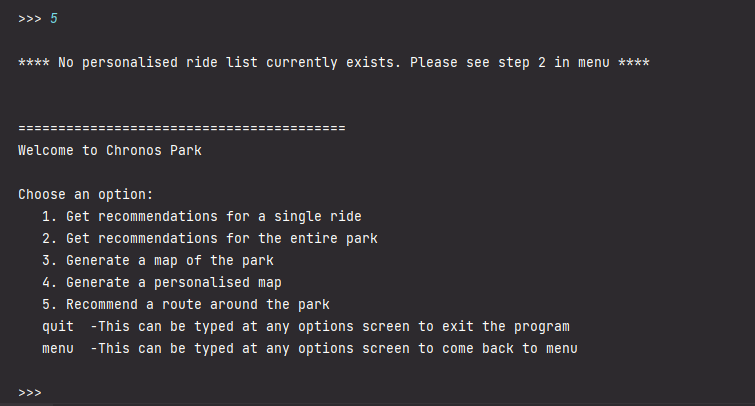
printing properly



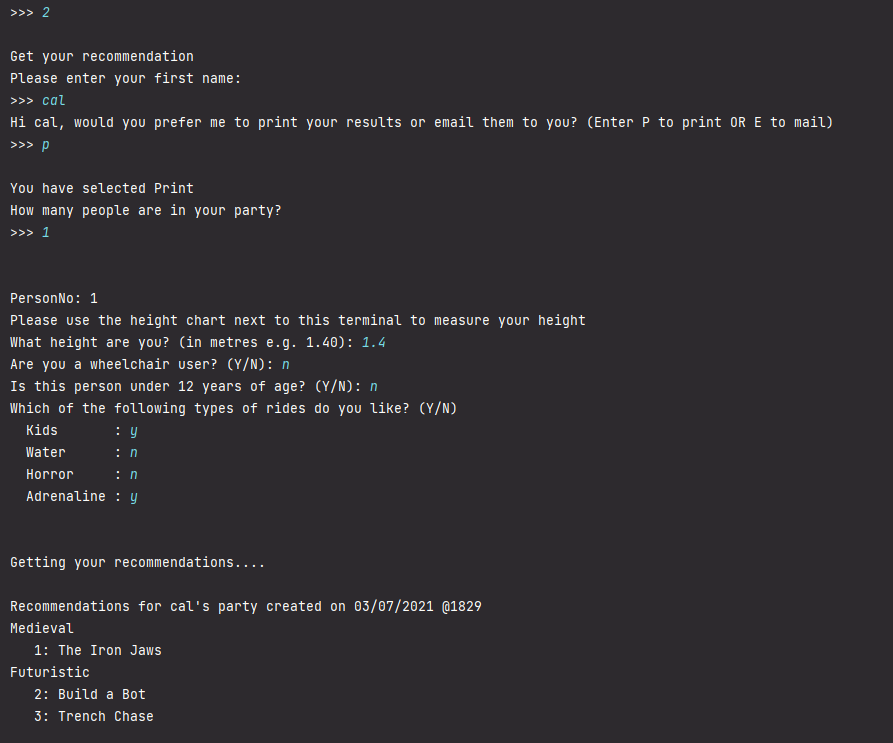
# Testing & Evidence – Step4

## Data outputs

no personalised input (part 2 yet)



testdata



printing properly

